Vorhersagequalität zufälliger Baumstrukturen

Vorhersagequalität zufälliger Baumstrukturen
Bachelor-Thesis von Alexander Heinz
März 2011

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Fachgebiet Knowledge Engineering

Betreuer & Prüfer:
Prof. Dr. Johannes Fürnkranz

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 08.03.2011

_____

(Alexander Heinz)

Einleitung

In einer Gesellschaft, in der immer mehr Daten gesammelt und gespeichert werden, gewinnen Methoden zur Auswertung dieser Daten immer weiter an Bedeutung. Wächst eine Datensammlung auf eine solche Größe, dass ein menschlicher Experte mit der Auswertung überfordert wäre, sind Computerprogramme aus dem Bereich Data Mining notwendig, um diese Aufgabe zu übernehmen. Zu diesen Programmen zählen Algorithmen zur Induktion von Entscheidungsbäumen. Sie arbeiten auf Datensätzen, die aus Ansammlungen von Instanzen bestehen, welche wiederum als Menge von Attributwerten bezüglich festgelegter Attribute definiert sind. Ein bestimmtes Attribut nimmt dabei die Position des Klassenattributes ein, welches jede Instanz einer Klasse zuordnet. Der Algorithmus erstellt auf Basis eines Trainingssets aus Instanzen einen Entscheidungsbaum, dessen Aufgabe es ist, unklassifizierte Instanzen anhand ihrer Attributwerte möglichst korrekt einer Klasse zuzuordnen. Dazu wird an jedem inneren Knoten des Entscheidungsbaumes ein Test bezüglich eines Attributes durchgeführt. Jedes Blatt beinhaltet eine Prognose, welcher Klasse die Instanzen, die es erreichen, angehören.
Viele Algorithmen zur Induktion von Entscheidungsbäumen verfahren nach einem ähnlichen Prinzip: Sie wählen zunächst ein Attribut, welches die zugrunde liegenden Daten als Wurzelknoten auftrennt. Rekursiv wird dieser Vorgang dann auf jedem Kindsknoten mit den jeweiligen Teilen der aufgetrennten Daten wiederholt, bis schließlich nicht mehr genügend Daten für weitere Kindsknoten vorhanden sind, oder eine andere Abbruchbedingung erfüllt wird. Der signifikanteste Unterschied dieser Algorithmen findet sich in den Kriterien der Attributwahl, die an jedem Knoten stattfindet. Um diese Kriterien zu bewerten, wäre es von Vorteil, sie mit einer Art neutralem Benchmark zu vergleichen, mithilfe dessen man die Effizienz der jeweiligen Kriterien messen könnte. Zu diesem Zweck wurde im Rahmen dieser Bachelorarbeit der Algorithmus RandomJ48 entwickelt. Es handelt sich dabei um eine Abwandlung des Algorithmus C4.5, dessen Attributwahl so modifiziert wurde, dass sie zufällig erfolgt.
Zunächst wird der Algorithmus C4.5 in seiner unveränderten Form vorgestellt, und an seinem Beispiel die zugrunde liegenden Konzepte bei der Wahl eines Attributs erläutert. Auf Basis dieses Wissens folgt eine Erklärung des Algorithmus RandomJ48 und der Designentscheidungen, die seine Entwicklung beeinflusst haben. Danach werden die Experimente, welche zum Vergleich der beiden Algorithmen durchgeführt wurden, erläutert. Schließlich folgt eine Auswertung der Ergebnisse dieser Experimente und Anregungen zu weiterführenden Forschungen auf diesem Gebiet.

1. Inhaltsverzeichnis

1. Inhaltsverzeichnis 4

2. C4.5/J48 5

2.1. Occam's Razor 5

2.2. Entropie 7

2.3. Information Gain und Gain Ratio 8

2.4. Numerische Attribute 10

2.5. Fehlende Attributwerte 12

2.6. Overfitting 13

2.7. Pruning 16

3. RandomJ48 18

3.1. Numerische Attribute 18

3.2. Variable Baumgröße 19

4. Experimente zum Vergleich der Algorithmen 21

4.1. Ohne Pruning 22

4.1.1. Baumgröße 22

4.1.2. Performanz 23

4.2. Mit Pruning 24

4.2.1. Baumgröße 24

4.2.2. Performanz 24

4.3. Auswirkung des Prunings auf die Baumgröße 25

4.4. Auswirkung des Prunings auf die Performanz 26

4.5. Zusammenfassung 28

5. Ausblick 29

Literaturverzeichnis 30

Anhang 31

2. C4.5/J48

Der Algorithmus C4.5 wurde 1992 von Ross Quinlan nach 14 Jahren Entwicklungszeit als Nachfolger des ebenfalls von ihm stammenden Algorithmus ID3 veröffentlicht ([QUI93] S. vii). Er gilt als einer der bekanntesten und renommiertesten Algorithmen zur Induktion von Entscheidungsbäumen. So schreiben Witten und Frank: "Although others have worked on similar methods, Quinlan's research has alway been at the very forefront of decision tree induction."([WIT05] S. 105). Dies macht ihn zu einer guten Wahl als Grundlage des in dieser Arbeit entwickelten Algorithmus RandomJ48.

Bei J48 handelt es sich um eine Java Implementierung des Algorithmus C4.5. Wenn in dieser Arbeit vom "Algorithmus J48" gesprochen wird, ist damit C4.5 in der Form, wie er in der Data Mining Software *Weka*[1] implementiert ist, gemeint. Um seinen Aufbau zu verstehen, sollten zunächst einige Grundkonzepte erläutert werden. Ein Großteil der dabei verwendeten Beispiele basiert auf folgendem Datensatz:

Tagesart	Feiertag	Uhrzeit	Offen
Wochentag	Nein	8.48	Ja
Wochentag	Nein	11.53	Nein
Wochentag	Nein	13.37	Ja
Wochentag	Nein	4.05	Nein
Wochentag	Nein	9.41	Ja
Wochentag	Nein	14.52	Ja
Wochentag	Nein	16.15	Ja
Wochentag	Nein	15.31	Ja
Wochentag	Ja	8.23	Nein
Wochenende	Nein	10.42	Nein
Wochenende	Nein	5.19	Nein
Wochenende	Ja	10.20	Nein

Tabelle 1: Beispiel-Datenset "Öffnungszeiten"

Dieser Datensatz besteht aus Stichproben zur Öffnungszeit eines Geschäftes. Bei jeder Instanz wird dabei festgestellt, ob es sich um einen Wochentag oder Wochenende handelt, ob dieser Tag ein Feiertag ist oder nicht, zu welcher Uhrzeit diese Stichprobe stattfand, und schließlich, ob das Geschäft zu diesem Zeitpunkt geöffnet oder geschlossen ist. Trotz der Einschränkungen der möglichen Attributwerte des Attributes *Uhrzeit* durch die Konventionen der Zeitrechnung wird es als kontinuierliches numerisches Attribut behandelt. Das Attribut "Offen" stellt das Klassenattribut dar.

2.1. Occam's Razor

Occam's Razor ist eine den Aufbau von J48 beeinflussende Heuristik. Mitchell definiert ihre Grundaussage als "Prefer the simplest hypothesis that fits the data." ([MIT97], S. 65). Sie sagt folglich aus, dass unter allen Hypothesen, die die Daten erklären, diejenige bevorzugt werden sollte, welche die wenigsten Annahmen macht. Dies lässt sich leicht auf die Domäne der Entscheidungsbäume übersetzen, wenn man sich vor Augen führt, dass ein Entscheidungsbaum gleichbedeutend einer Menge an Regeln ist:

[1] Für weitere Information zu Weka siehe Kapitel 4

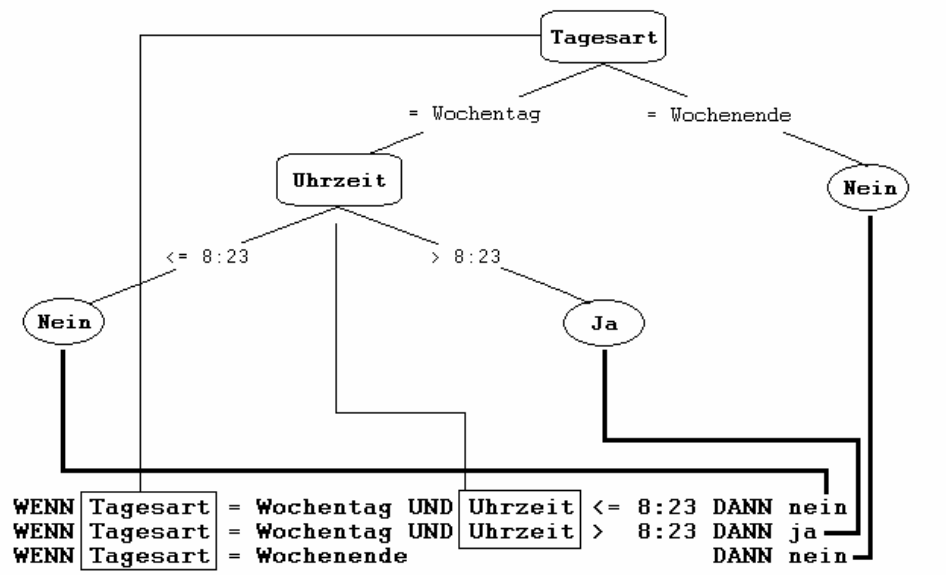


Abbildung 1: Ein Entscheidungsbaum und das zugehörige Set an Regeln

Jede Regel repräsentiert dabei einen Weg von der Wurzel zu einem Blatt. Jeder Knoten des Entscheidungsbaums fügt den Regeln, welche die Wege darstellen, die durch diesen Knoten verlaufen, jeweils eine weitere Annahme hinzu, wie hier an den Knoten *Tagesart* und *Uhrzeit* zu erkennen ist. Folglich steht jeder Knoten für Annahmen, die von der Regelmenge und damit vom Entscheidungsbaum gemacht werden. Weniger Knoten stehen für weniger Annahmen, wie man beispielsweise durch das Entfernen des Knotens *Uhrzeit* sieht:

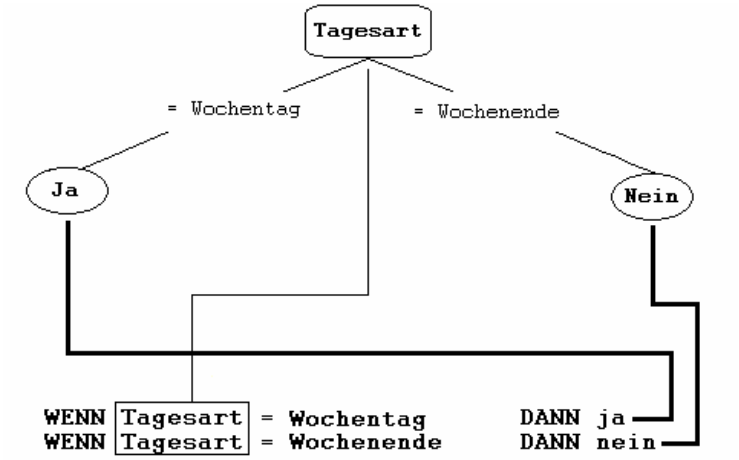


Abbildung 2: Baum und Regelset nach Entfernen des Knotens *Uhrzeit*

Durch das Entfernen des Knotens *Uhrzeit* wurden auch die Annahmen des Regelsets bezüglich dieses Attributes entfernt, was das Regelset vereinfacht. Will man also *Occam's Razor* befolgen und einen Entscheidungsbaum mit möglichst wenigen Annahmen erhalten, ist ein Baum mit möglichst wenigen Knoten anzustreben.

2.2. Entropie

Das Streben nach einem möglichst kleinen Baum ist ein maßgebender Faktor bei der Entscheidung, welches Attribut zum Auftrennen der Daten ("Splitten") gewählt werden sollte. Im Idealfall lässt sich ein Attribut finden, welches die diesen Knoten erreichenden Instanzen gemäß ihrer Klassenzugehörigkeit teilt, in dem Sinne, dass für jede vorhandene Klasse ein Kindsknoten erstellt wird, den jeweils alle Instanzen dieser bestimmten Klasse erreichen. Da an allen Kindsknoten dann nur noch Instanzen jeweils einer Klasse vorhanden wären und eine weitere Aufteilung damit sinnlos wäre, würden sie zu Blätter umfunktioniert und der Baum wäre an dieser Stelle fertig erstellt. Dies ist jedoch nur in seltenen Fällen möglich. Oft muss man sich stattdessen mit einer Annäherung an diesen Zustand begnügen und ein Attribut finden, das die Instanzen möglichst "rein" anhand ihrer Klassenzugehörigkeiten aufteilt, also jedem Kindsknoten möglichst viele Instanzen einer Klasse und möglichst wenige der anderen Klassen zuordnet. Als Maß dieser Reinheit wird die Entropie verwendet[1]. In der Domäne der Entscheidungsbäume steht sie für "die minimale Anzahl an Bits, die nötig sind, um die Klassifikationsgenauigkeit einer Instanz zu codieren."([ALP08] S. 186). Berechnet wird sie folgendermaßen:

P_i: Anteil der Instanzen im Datensatz, die zur Klasse i gehören

$$\textbf{Entropie}(P_1, P_2, P_3, \ldots P_n) = -\sum_{i=1}^{n} P_i * \log_2(P_i) \text{ bits}$$

Zur Veranschaulichung betrachten wir zunächst das Beispiel-Datenset[2]. Da bei diesem Datensatz das Klassenattribut *Offen* zwei mögliche Werte besitzt, existieren zwei Klassen und wir führen folgende Notation ein:

X+: X positive Instanzen (Offen = Ja)
X–: X negative Instanzen (Offen = Nein)
Entropie([X+, Y-]) : Entropie einer Verteilung mit X positiven und Y negativen Instanzen

Nun lässt sich die Entropie des gesamten Datensatzes berechnen:

Entropie(Datensatz) = Entropie([6+, 6-]) = – (6/12) log(6/12) bits – (6/12) log(6/12) bits = 1 bit

Bei einem Split werden die Entropien der entstehenden Sets anteilmäßig addiert:

Entropie(Tagesart) = (9/12) Entropie[6+, 3-] + (3/12) Entropie[0+, 3-]
= (9/12) 0,92 bits + (3/12) 0 bits
= 0,69 bits

Entropie(Feiertag) = (10/12) Entropie[6+, 4-] + (2/12) Entropie[0+ 2-]
= (10/12) 0,97 bits + (2/12) 0 bits
= 0,81 bits

Da es sich bei *Uhrzeit* um ein numerisches Attribut handelt, werden zur Bestimmung der Entropie der entstehenden Verteilung nach einem Split an diesem Attribut weitere Berechnungen benötigt, mit denen sich Kapitel 2.4 befasst

[1] Es wären auch andere Maße möglich, z.B. der Gini-Index oder der Fehlklassifikationfehler, siehe [ALP08] S.187f.

[2] Siehe Kapitel 2

2.3. Information Gain und Gain Ratio

Als *Information Gain* eines Attributs wird die Differenz zwischen der aktuellen Entropie der Instanzen am gerade betrachteten Knoten und der Entropie der entstehenden Verteilung bei einem Split durch dieses Attribut bezeichnet:

Information Gain(Tagesart) = Entropie(Datensatz) – Entropie(Tagesart)
= 1 bit – 0,69 bits
= 0,31 bits

Information Gain(Feiertag) = Entropie(Datensatz) – Entropie(Feiertag)
= 1 bit – 0,81 bits
= 0,19 bits

Da wir einen möglichst kleinen Baum anstreben, scheint es sinnvoll zu sein, bei einem Split dasjenige Attribut auszuwählen, welches den *Information Gain* maximiert, um so schnell zu reinen Kindsknoten zu gelangen, die daraufhin als Blätter fungieren. Dieses Vorgehen minimiert allerdings nicht zwingend die Komplexität des entstehenden Entscheidungsbaumes. Wie Quinlan bemerkte, werden dabei Attribute mit hoher Anzahl an möglichen Attributwerten bevorzugt ([QUI93]: S. 23). Angesichts der Tatsache, dass der Anzahl an möglichen Werten eines Attributes keine Grenzen gesetzt sind, würde beispielsweise die Wahl eines Attributes, welches für jede Instanz einen eigenen Wert besitzt, zwar immer den maximalen *Information Gain* produzieren, aber eine relativ hohe Komplexität erzeugen. Ein Beispiel eines solchen Attributes im Rahmen des bisher verwendeten Datensatzes *Öffnungszeiten* wäre ein Attribut *Datum*, dessen Wert für jede Instanz unterschiedlich wäre. Ein Split an diesem Attribut zu Beginn des Entstehungsprozesses des Entscheidungsbaumes würde zu 12 Kindsknoten führen, die jeweils eine Entropie von 0 besäßen. Die Entropie dieses Splits wäre damit ebenfalls 0 und würde den Information Gain maximieren. Der entstehende Baum wäre aber zum einen unnötig komplex, und würde zum anderen bei der Klassifizierung neuer Instanzen keine gute Figur machen, da diese mit hoher Wahrscheinlichkeit neue Attributwerte des Attributes *Datum* einführen würden.

Um dieses Problem zu beheben, entwickelte Quinlan die *Gain Ratio*, welche die Anzahl und Größe der entstehenden Kindsknoten in die Wahl eines Attributs für den Split miteinbezieht:

Gain Ratio = Information Gain/Split Information

Die Gain Ratio teilt den Information Gain durch die sogenannte Split Information. Diese errechnet sich ähnlich der Entropie, statt aus Häufigkeiten der Klassenzugehörigkeit bestehen die Argumente hier jedoch aus den Häufigkeiten der den Kindsknoten zugeteilten Instanzen:

Q_i: Anteil der Instanzen, die dem Kindsknoten i zugeteilt werden

$$\textbf{Split Information}(Q_1, Q_2, Q_3, \ldots, Q_n) = -\sum_{i=1}^{n} Q_i * \log_i(Q_i)$$

Bei einem Split am Attribut *Tagesart* würden die Werte also folgendermaßen aussehen:

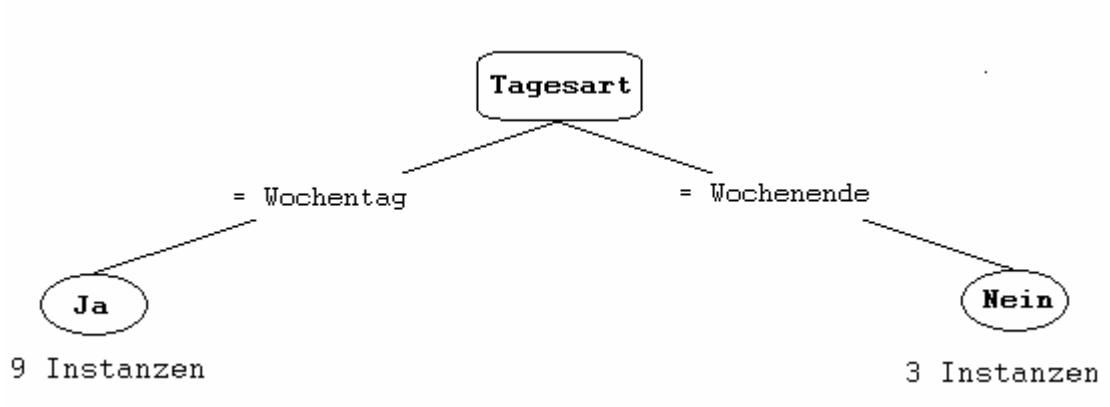


Abbildung 3: Split an Attribut *Tagesart*

Split Information(Tagesart) = -9/12 * $\log_2$ (9/12) bits - 3/12 * $\log_2$ (3/12) bits = 0,81 bits

Damit lassen sich nun alle Werte errechnen, die Einfluss auf die Attributwahl haben:

Entropie(Öffnungszeiten) = 1 bit
Entropie(Tagesart) = 0,69 bits
Information Gain(Tagesart) = 1 bit – 0,69 bits= 0,31 bits
Gain Ratio(Tagesart) = 0,31 bits/ 0,81 bits = 0,3827

Analog würde ein Split am Attribut *Feiertag* folgende Werte ergeben:

Abbildung 4: Split am Attribut *Feiertag*

Split Information(Feiertag) = -10/12 * $\log_2$ (10/12) bits - 2/12 * $\log_2$(2/12) bits = 0,65 bits

Entropie(Öffnungszeiten) = 1 bit
Entropie(Feiertag) = 0,8083 bits
Information Gain(Feiertag) = 1 bit – 0,81 bits = 0,19 bits
Gain Ratio(Feiertag) = 0,19 bits / 0,65 bits = 0,2923

Da die Gain Ratio von *Tagesart* mit 0,3827 die von *Feiertag* mit 0,2923 übertrifft, würde in diesem Fall J48 *Tagesart* als Split-Attribut den Vorzug geben, solange keine weiteren Attribute mit höherer Gain Ratio vorhanden sind.

2.4. Numerische Attribute

Bei nominellen Attributen genügt deren Auswahl bei einem Split, um diesen durchzuführen und die Kindsknoten zu erstellen. Bei numerischen Attributen muss zunächst eine weitere Frage beantwortet werden: An welchen Stellen der Bandbreite aller möglichen Attributwerte sollen die Instanzen aufgetrennt werden?

Eine Möglichkeit wäre, das Attribut vor der Erstellung des Entscheidungsbaumes zu diskretisieren. Am Beispiel des Attributes *Uhrzeit* würde dies bedeuten, dass man den Wertebereich nicht mehr als kontinuierlich ansieht, sondern in Bereiche unterteilt. So könnte es die Werte {0:00 – 0:59, 1:00 – 1:59, 2:00 – 2:59, ..., 23:00 – 23:59} annehmen. Auch eine gröbere Aufteilung in {morgens, mittags, abends, nachts} ist denkbar. Damit wäre "Uhrzeit" in ein nominelles Attribut umgewandelt.

Dies ist allerdings oft nicht praktikabel, da die Ersteller eines Datensatzes eine solche Diskretisierung, so sie denn sinnvoll wäre, bereits hätten durchführen können. Die Alternative besteht darin, bei jeder Wahl eines nominellen Attributes Werte zu finden, die den Wertebereich dermaßen auftrennen, dass eine möglichst hohe Gain Ratio entsteht. J48 teilt numerische Attribute binär, wobei eine Instanz als eine Art Trennwand des Wertebereichs ausgewählt wird. Diese Instanz und alle mit niedrigerem Wert bezüglich des Attributes werden dem linken Kindsknoten zugeteilt, alle Instanzen mit größerem Wert dem rechten. Würde beispielsweise das Attribut *Uhrzeit* ausgewählt, sähe die Verteilung der Attributwerte folgendermaßen aus:

\+ : positive Instanz
– : negative Instanz

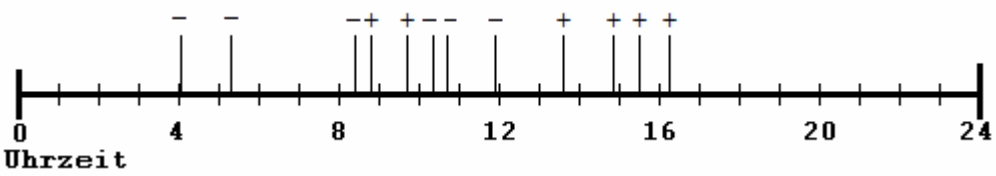


Abbildung 5: Verteilung der Instanzen im Wertebereich des Attributes *Uhrzeit*

Die Anzahl positiver und negativer Instanzen der entstehenden Kindsknoten (links: linker Kindsknoten, rechts: rechter Kindsknoten) sowie die jeweils entstehende Gain Ratio durch die Wahl einer bestimmten Zeit als Grenzwert sind folgender Tabelle zu entnehmen:

Wahl der Grenze	# Positiv links	# Negativ links	# Positiv rechts	# Negativ rechts	Gain Ratio
4.05	0	1	6	5	0,2166
5.19	0	2	6	4	0,2937
8.23	0	3	6	3	0,3843
8.48	1	3	5	3	0,1025
9.41	2	3	4	3	0,0213
10.20	2	4	4	2	0,0817
10.42	2	5	4	1	0,2018
11.53	**2**	**6**	**4**	**0**	**0,5045**
13.37	3	6	3	0	0,3843
14.52	4	6	2	0	0,2937
15.31	5	6	1	0	0,2166

Tabelle 2: Entstehende Verteilungen und Gain Ratio bei Wahl einer Zeit als Grenze

Die höchste Gain Ratio entsteht bei einem Split bei 11.53, wobei die Instanzen folgendermaßen aufgeteilt werden:

\+ : positive Instanz
– : negative Instanz

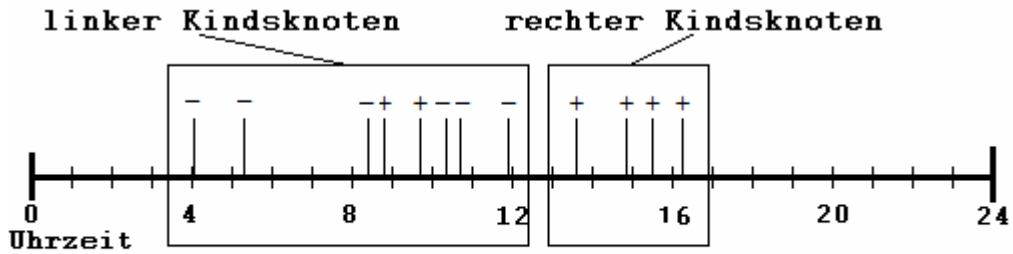


Abbildung 6: Aufteilung der Instanzen auf die Kindsknoten bei Split an 11:53

Weiterhin sei noch anzumerken, dass durch die Auswahl an möglichen Splitpoints eines numerischen Attributes mehrere sinnvolle Splits dieses Attributs innerhalb eines Entscheidungsbaumes möglich sind, während bei nominellen Attributen ein zweiter Split keinen positiven Information Gain produzieren kann.

2.5. Fehlende Attributwerte

Viele Datensätze bestehen aus fehlerhaften Daten, bei denen z.B. aufgrund von Messungenauigkeiten Werte nicht aufgezeichnet wurden. Um Instanzen mit fehlenden Attributwerten bei der Entstehung eines Entscheidungsbaumes miteinzubeziehen, versieht J48 jede Instanz mit einem Gewicht, welches zunächst auf 1 initialisiert wird. Bei der Wahl eines Attributes, dessen Wert bei einer Instanz fehlt, wird diese nun aufgeteilt und an sämtliche Kindsknoten weitergegeben. Die Gewichte dieser Teilinstanzen orientieren sich dabei daran, wie viele weitere Instanzen an den jeweiligen Kindsknoten weitergeschickt werden. Zur Veranschaulichung sei folgende Instanz zu betrachten:

Wochentag	Nein	16.15	Ja

Ihr Weg während des Entstehungsprozesses des Entscheidungsbaumes und das dabei mitgeführte Gewicht sehen folgendermaßen aus:

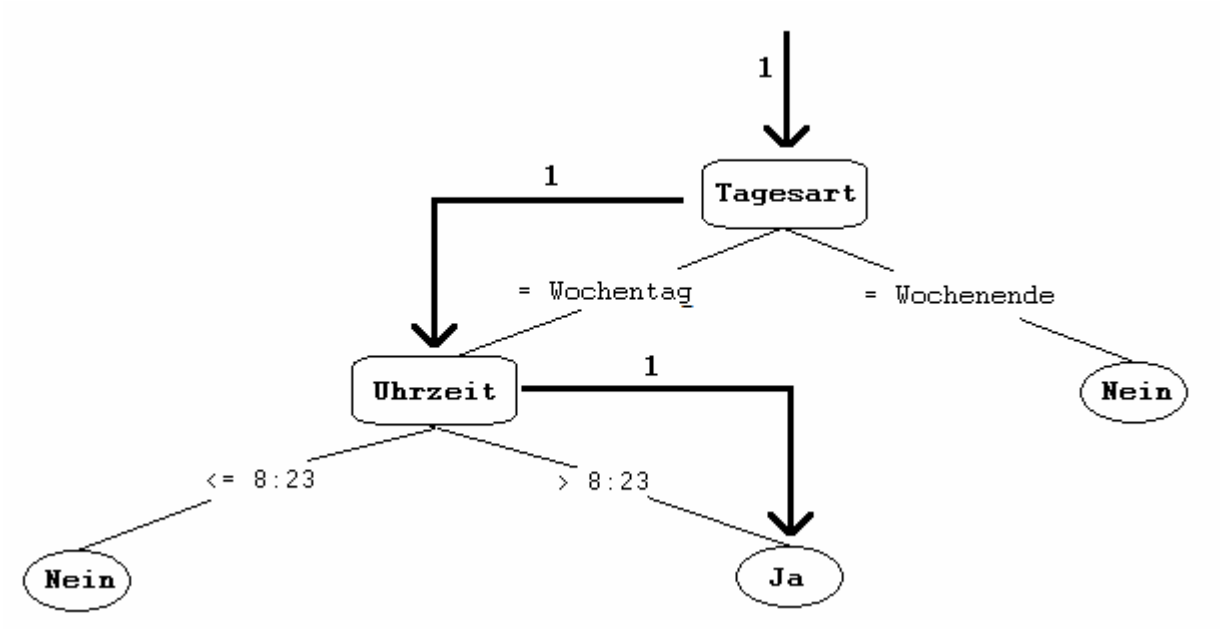


Abbildung 7: Weg der vollständigen Instanz durch den Baum

Die Instanz behält während des gesamten Vorgangs ihr Gewicht von 1. Wenn wir die Instanz allerdings umformen zu

?	Nein	16.15	Ja

wobei ein Fragezeichen für einen fehlenden Wert steht, führt ein Split an diesem fehlenden Wert zu einer Aufteilung dieser Instanz:

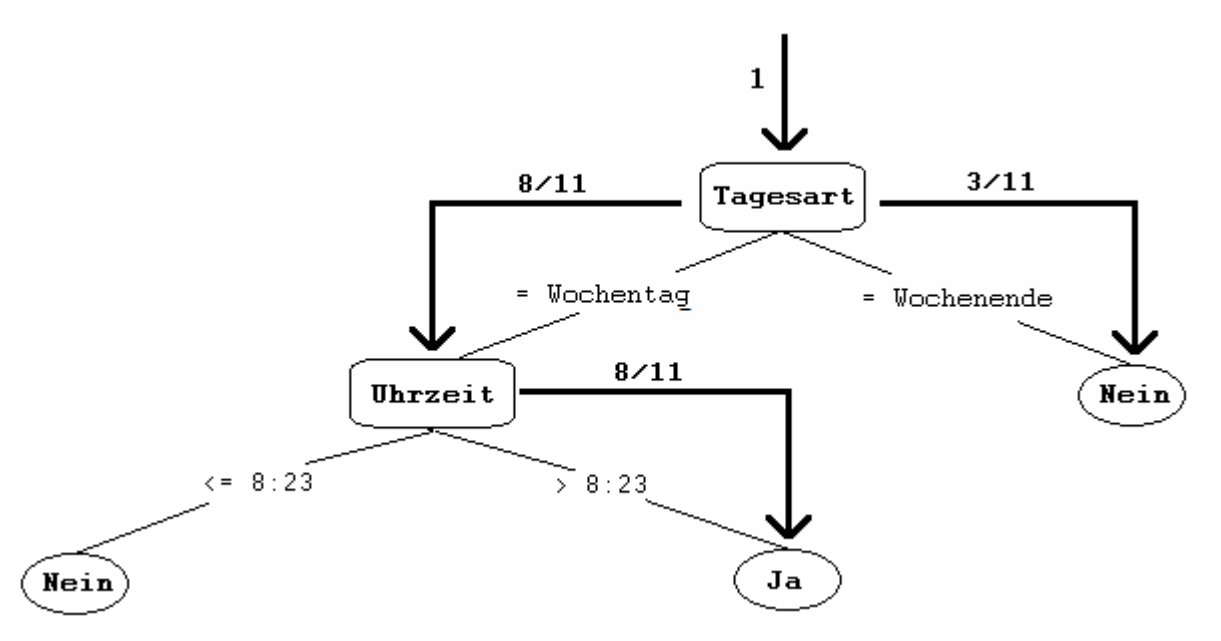


Abbildung 8: Weg der Teilinstanzen durch den Baum

Die Wurzel teilt die 11 Instanzen, bei denen keine Werte fehlen, in 8 für den linken und 3 für den rechten Kindsknoten auf. Die von uns betrachtete Instanz, bei der der Wert des Attributes *Tagesart* fehlt, wird somit zu 8/11 dem linken und zu 3/11 dem rechten Kindsknoten zugeteilt. Würde auch der Wert des Attributes *Uhrzeit* fehlen, würde eine weitere Aufteilung der Teilinstanz stattfinden.

2.6. Overfitting

Overfitting bezeichnet ein zu starkes Anpassen des Entscheidungsbaumes an die Instanzen, die an seinem Entstehungsprozess beteiligt sind. Es tritt dann auf, wenn das Trainingsset fehlerhafte Instanzen und damit Rauschen beinhaltet, insbesondere wenn das Trainingsset relativ klein ist und die fehlerhaften Instanzen somit bei der Entstehung des Baumes mehr ins Gewicht fallen. Je komplexer ein Entscheidungsbaum ist, desto wahrscheinlicher ist es, dass er sich an das Rauschen anpasst (vgl. [ALP08] S. 36). Die Folge ist eine schlechtere Performanz des Baumes bei der Klassifizierung neuer Instanzen.

Zur Veranschaulichung betrachten wir einen zweidimensionalen Raum. Es soll ein Entscheidungsbaum erstellt werden, der bestimmt, ob sich ein Punkt innerhalb oder außerhalb des in folgender Abbildung dargestellten Quadrats befindet.

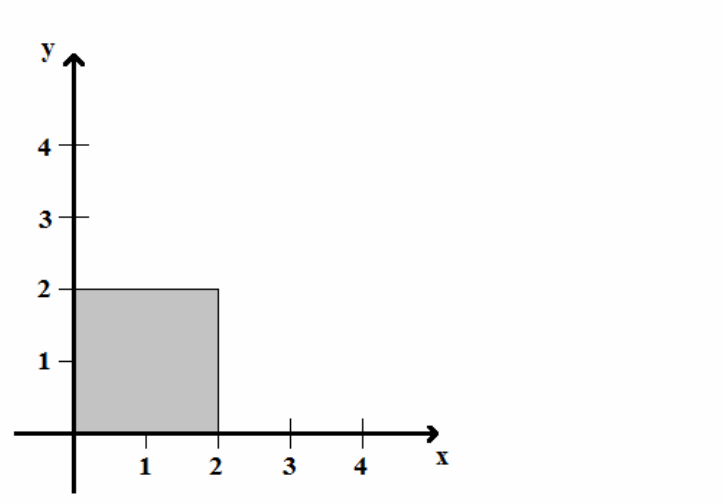


Abbildung 9: Zu lernendes Konzept

Als Trainingset seien folgende Instanzen gegeben, wobei die letzte Instanz ein falsches Beispiel und damit Rauschen darstellt:

x	y	Klasse (N = Negativ, P = Positiv)
1,5	0,5	P
1,5	1,5	P
2,5	1	N
1,5	2,5	N
0,5	1	N

Tabelle 3: Trainingsset des Quadrat Problems

Ein Algorithmus, dem diese Instanzen als Trainingset dienen, könnte beispielsweise den folgenden Baum erstellen:

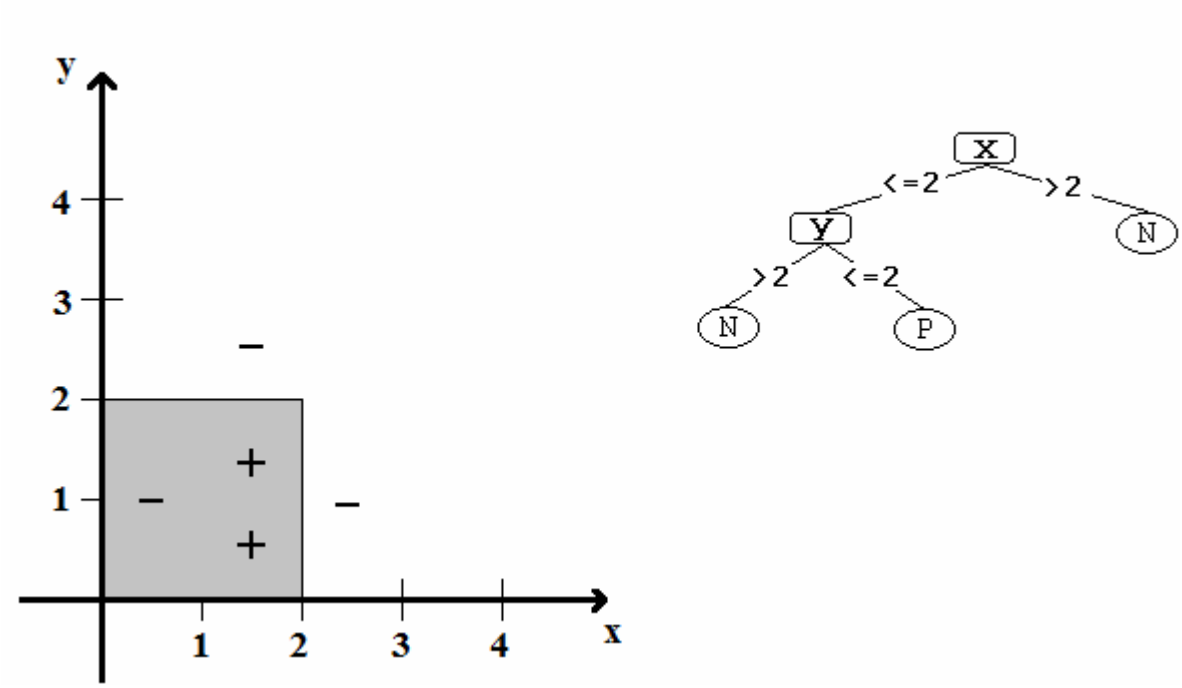


Abbildung 10: Einfacher Entscheidungsbaum zum Quadrat Problem

Es ist aber auch möglich, dass sich der Klassifizierer weiter an die Daten anpasst und der Baum erweitert wird, sodass er komplexer wird und folgendermaßen aussieht:

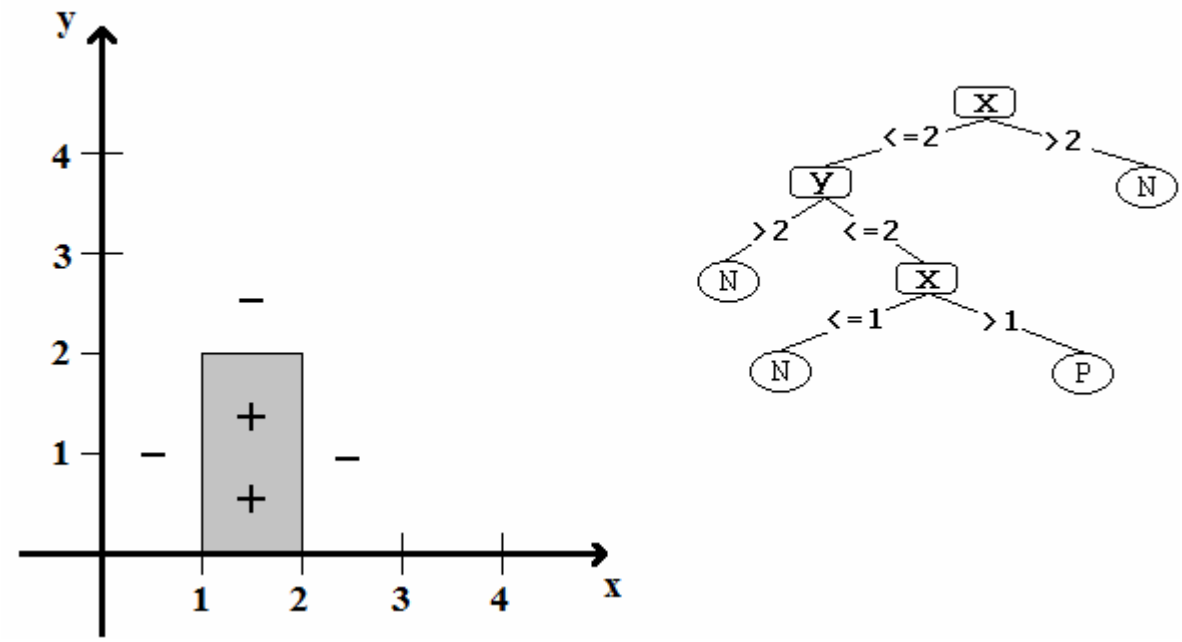


Abbildung 11: Komplexerer Entscheidungsbaum zum Quadrat Problem

Der komplexere Baum klassifiziert alle Trainingsinstanzen korrekt und ist in diesem Bereich dem einfacheren Baum überlegen, schneidet aber beim Klassifizieren neuer Instanzen schlechter ab. Er ist "overfitted".

Bei Experimenten mit ID3 auf rauschbehafteten Datensätzen sorgte Overfitting für eine Verringerung der Genauigkeit der Entscheidungsbäume um 10 - 25% bei den meisten Problemen(Mingers 1989, zit. bei [MIT97], S.68). Selbst bei rauschfreien Datensätzen kann es zu Overfitting kommen (vgl. [MIT97] S. 68). Es erscheint also sinnvoll, den entstanden Baum auf Overfitting zu überprüfen, und gegebenenfalls die Komplexität zu verringern. Dies geschieht beim Pruning.

2.7. Pruning

Pruning bezeichnet das Reduzieren von Annahmen, die ein Klassifizierer macht, um Overfitting zu vermeiden und damit die Performanz auf unbekannten Daten zu verbessern. Im Falle von Entscheidungsbäumen bedeutet dies, Knoten zu entfernen. J48 verwendet Postpruning, d.h. das Pruning wird nach der Entstehung des Baumes vollzogen, im Gegensatz zu Prepruning, bei dem es währenddessen stattfindet.

In J48 arbeitet sich Pruning von den Blättern zur Wurzel vor, und entscheidet dabei an jedem inneren Knoten, ob der von ihm ausgehende Teilbaum ersetzt werden soll. Dabei gibt es zwei Ersetzungsstrategien. Die erste nennt sich *subtree-replacement*, und ersetzt einen Teilbaum durch ein Blatt, hier zu sehen beim Teilbaum, dessen Wurzel der Knoten *Uhrzeit* darstellt:

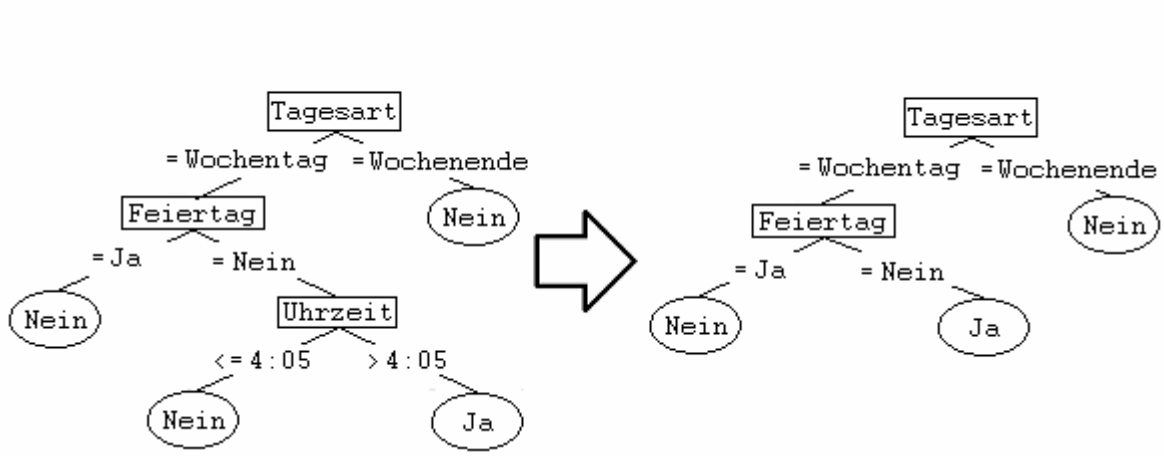


Abbildung 12: Subtree-replacement

Bei der zweiten Strategie handelt es sich um *subtree-raising*, wobei ein Teilbaum durch einen tieferen ersetzt wird, wie hier der Teilbaum mit Wurzelknoten *Feiertag* durch denjenigen mit Wurzelknoten *Uhrzeit*:

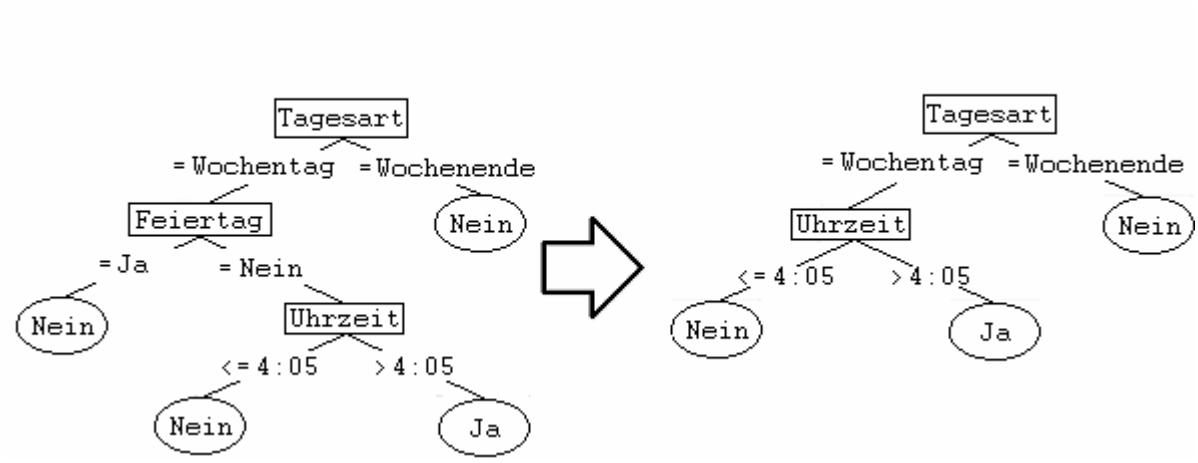


Abbildung 13: Subtree-raising

Ob ein Teilbaum durch ein Blatt oder einen tieferen Teilbaum ersetzt werden soll, entscheiden dabei statistische Verfahren, welche anhand der Fehlerrate eines Teilbaums auf den Trainingsdaten Vorhersagen über die voraussichtliche Fehlerrate des Teilbaums auf der gesamten Population an Daten berechnen.[1]

[1] Detaillierte Erklärungen dieser Verfahren finden sich in [QUI93], S. 37-43.

3. RandomJ48

RandomJ48 ist eine modifizierte Version des beschriebenen J48 Algorithmus. Sein Ziel besteht darin, einen Entscheidungsbaum auf die gleiche Weise wie J48 zu erstellen, mit dem Unterschied, dass an jedem Knoten die Wahl des Attributes, an dem gesplittet werden soll, zufällig erfolgt. Durch diese Zufallswahl fallen die Berechnungen zu Information Gain, Split Information und Gain Ratio weg. Das Vorgehen bei fehlenden Attributwerten und die Durchführung des Prunings verändern sich gegenüber J48 nicht.

3.1. Numerische Attribute

Wie bei J48 ist es auch bei RandomJ48 erforderlich, nach Auswahl eines numerischen Attributes einen Wert zu bestimmen, welcher die Instanzen in zwei Bereiche aufteilt. In diesem Fall unterliegt die Wahl jedoch nicht der Forderung nach einer maximalen Gain Ratio, sondern soll zufällig erfolgen. Hierbei gibt es grundsätzlich zwei Ansätze.

Ein mögliches Vorgehen besteht darin, den gesamten Wertebereich an Attributwerten zu betrachten, und einen Zufallswert zwischen Minimum und Maximum dieses Bereiches[1] auszuwählen. Es wäre aber auch denkbar, aus den vorhandenen Instanzen per Zufall eine auszuwählen und diese als Grenze zu bestimmen. Der Unterschied besteht darin, dass der zweite Ansatz sensitiv gegenüber der Verteilung der Instanzen im Wertebereich ist: Bereiche, denen mehr Instanzen zugeordnet sind, werden bei diesem Ansatz mit höherer Wahrscheinlichkeit als Splitpoint ausgewählt.

Da ein Vergleich mit J48 angestrebt wird, sollte man sich an dessen Implementierung orientieren. Auf den ersten Blick erscheint der zweite Ansatz passender, da auch J48 jede Instanz in Betracht zieht, und schließlich diejenige mit dem höchsten Information Gain auswählt. Allerdings ist es durchaus möglich, dass theoretisch jeder Wert im potenziellen Wertebereich als Splitpoint möglich wäre, um die Effizienz zu erhöhen aber nur die Attributwerte der vorhandenen Instanzen getestet werden, da der gesamte Bereich zwischen den Werten zweier Instanzen die Instanzen auf gleiche Weise einteilen würde und damit eine identische Gain Ratio produzieren würde, wie folgende Grafik verdeutlicht:

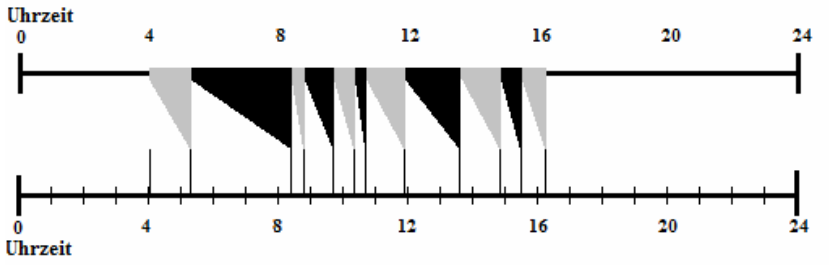


Abbildung 14: Bereiche mit gleichem Information Gain

Auf der unteren Zeitleiste sind die 12 Instanzen des *Öffnungszeiten* Datensatzes bezüglich ihrer Werte des Attributes *Uhrzeit* markiert. Die obere Zeitleiste zeigt Bereiche an, bei deren Wahl der gleiche Information Gain entstünde wie bei der Wahl der mit dem jeweiligen Bereich verknüpften Instanz. Es sind also beide Varianten denkbar. Im Rahmen dieser Bachelorarbeit wurde sich für die Wahl einer zufälligen Instanz entschieden.

[1] Aufgrund der Vorgabe, keine leeren Kindsknoten zu erzeugen, wäre hierbei nur der Bereich vom kleinsten vorkommenden Attributwert bis zum größten relevant, wobei der größte Attributwert selbst ausgeschlossen ist.

3.2. Variable Baumgröße

J48 wird auf dem gleichen Datensatz bei jeder Ausführung den gleichen Entscheidungsbaum erstellen. Durch die zufällige Attributswahl ist dies bei RandomJ48 nicht der Fall. Um die Bäume der beiden Algorithmen vergleichbar zu machen, sollten diese jedoch die gleiche Größe haben. Es sollte also zunächst ein Verfahren entwickelt werden, um RandomJ48-Bäume einer bestimmten Größe zu erstellen.

Die einfachste Variante besteht darin, den Entstehungsprozess des Baumes bei einer festgelegten Tiefe abzubrechen. Aber auch RandomJ48-Bäume gleicher Tiefe weisen eine hohe Varianz der Anzahl ihrer Knoten auf, was man durch Stichproben schnell belegen kann:

Datensatz "anneal"[1], 66 % Trainingsset, 33 % Testset
10 mal RandomJ48 ohne Pruning mit eingestellter Baumtiefe 8

Stichprobe	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Knotenanzahl	117	165	103	101	95	105	206	153	91	60

Tabelle 4: Stichproben bei festgelegter Baumtiefe

Das Limitieren der Baumtiefe ist damit ungeeignet, um vergleichbare Bäume zu erstellen. Eine bessere Methode besteht darin, die geforderte Komplexität als Parameter beim Erstellen des Baumes mitzuführen, und bei jedem Split diesen Parameter im selben Verhältnis wie die Instanzen an die Kindsknoten weiterzugeben[2]. Beim Erstellen des Wurzelknotens eines Baumes, der auf dem Beispiel-Datensatz *Öffnungszeiten* beruht, und bei dem eine Komplexität von 5 gefordert wird, sähe dies folgendermaßen aus:

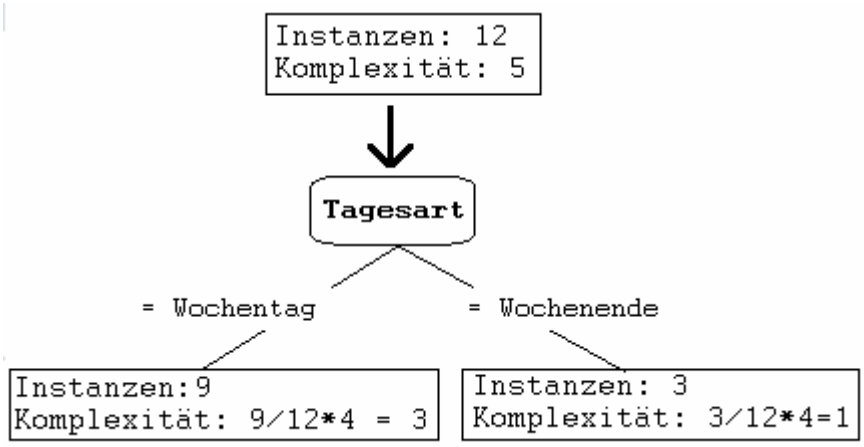


Abbildung 15: Auftrennung der geforderten Komplexität

Da die Wurzel selbst einen Knoten darstellt, wird sie von der Komplexität abgezogen. Von den verbleibenden Forderungen nach 4 Knoten werden drei an den linken, eine an den rechten Kindsknoten weitergeleitet.

Zu klären bleibt die Frage, ab welchem Wert aufgehört werden sollte, den Baum zu erweitern. Da RandomJ48 jedes am aktuell betrachteten Knoten wählbare Attribut mit gleicher Wahrscheinlichkeit

[1] Genauere Angaben zu diesem Datensatz finden sich in Kapitel 4.

[2] Hierbei wird ähnlich verfahren wie bei der Weitergabe von Teilinstanzen mit fehlenden Werten, siehe Kapitel 2.5

auswählt, wächst die Knotenanzahl des Baumes durch einen weiteren Split im Schnitt um den Durchschnitt der Anzahl der Attributwerte dieser Attribute[1], wobei numerische Attribute durch ihren binären Split als zwei Attributwerte zählen. Überschreitet die aufgerundete geforderte Komplexität diesen Schnitt, sollte ein Split durchgeführt werden.

Im Datensatz *Öffnungszeiten* befinden sich die nominellen Attribute *Tagesart* und *Feiertag* mit jeweils zwei Attributwerten, sowie das numerische Attribut *Uhrzeit*, das ebenfalls als zwei Attributwerte gilt. Ein Split an der Wurzel produziert damit im Schnitt (2 + 2 + 2) / 3 = 2 Kindsknoten. Die im vorgestellten Beispiel an der Wurzel geforderte Komplexität von 5, bzw. nach Abzug des Wurzelknotens selbst von 4, ist größer als der durchschnittliche Knotenzuwachs von 2, folglich sollte ein Split erfolgen.

Aber auch bei diesem Verfahren bestehen Probleme. Ist ein Knoten rein, besitzt er also nur Instanzen einer Klasse, dann wird an dieser Stelle das Baumwachstum beendet, ohne die geforderte Komplexität zu beachten, welche damit verloren geht. Ähnlich verhält es sich in einem fiktiven Beispiel, in dem 10 verschiedene Knoten jeweils eine Komplexität von eins fordern, während das durchschnittliche Baumwachstum bei 2 liegt. Auch bei diesem Baum gehen die Forderungen nach 10 weiteren Knoten verloren. Insgesamt erhalten wir bei dieser Methode lediglich eine Annäherung an die gewünschte Komplexität, wie auch folgende Stichproben zeigen:

Datensatz "anneal", 66 % Trainingsset, 33 % Testset
10-mal RandomJ48 ohne Pruning mit eingestellter Komplexität 72

Stichprobe	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Knotenanzahl	67	79	63	70	71	72	64	78	76	74

Tabelle 5: Stichproben bei mitgeführter Komplexität

Eine weitere Methode besteht darin, zunächst den Baum ohne Einschränkungen zu erstellen, und ihn danach durch Entfernen von Knoten "zurechtzuschneiden". Dieser Vorgang ist zwar rechenintensiver als die vorigen, da der Baum zunächst auf seine volle Größe wächst, er ermöglicht aber eine sehr genaue Anpassung an eine vorgegebene Komplexität: Es kann einfach Knoten für Knoten entfernt werden, bis die Größe des Baumes den Vorgaben entspricht.

Es bleibt zu klären, welche Knoten entfernt werden sollen. Um diejenigen Bäume zu erhalten, die entstanden wären, wenn das Baumwachstum nach Erreichen der geplanten Komplexität beendet worden wäre, wird sich hier für die Knoten entschieden, deren Kinder Blätter sind und die sich im jeweils größten Teilbaum befinden. Dies hat außerdem den Vorteil, dass diese Knoten am leichtesten zu entfernen sind, da der Baum bis auf die Umwandlung dieser Knoten zu Blätter nicht weiter umgeformt werden muss, wie es beispielsweise beim Entfernen eines Knoten mit Kindeskinderknoten der Fall wäre. Auch hier soll wieder eine Stichprobe die Qualität des Vorgehens bewerten:

Datensatz "anneal", 66% Trainingsset, 33% Testset
10-mal RandomJ48 ohne Pruning mit eingestellter Komplexität 72

Stichprobe	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Knotenanzahl	72	71	73	73	71	73	69	72	72	72

Tabelle 6: Stichproben beim Zurechtschneiden des Baumes

Diese Variante erreicht die präziseste Anpassung an die geforderte Komplexität der vorgestellten Methoden und wird daher von RandomJ48 verwendet.

[1] Da nominelle Attribute nicht mehrmals gewählt werden können, kann sich dieser Durchschnitt nach jedem Knoten ändern und muss neu berechnet werden.

4. Experimente zum Vergleich der Algorithmen

Um die beiden vorgestellten Algorithmen zu vergleichen, wurden Experimente auf zehn unterschiedlichen Datensätzen durchgeführt. Das Spektrum reicht hierbei von Datensätzen mit ausschließlich nominellen Attributen, mit ausschließlich numerischen Attributen, mit sowohl nominellen als auch numerischen Attributen bis hin zu Datensätzen mit unterschiedlichen Anteilen an fehlenden Attributwerten, damit ein möglichst repräsentativer Schnitt der in der Praxis vorkommenden Datensätze gewährleistet ist. Die verwendeten Datensätze stammen aus dem *UCI Machine Learning Repository*[1]. Folgende Datensätze wurden verwendet:

Datensatz	# Instanzen	# num. Attribute	# nom. Attribute	Fehlende Attribute
"anneal"	898	6	32	Viele
"horse-colic"	368	7	14	Einige
"flare"	1066	0	12	Keine
"kr-vs-kp"	3196	0	35	Keine
"mfeat"	2000	5	0	Keine
"segment"	5000	19	0	Keine
"sonar"	208	60	0	Keine
"synthetic"	600	60	0	Keine
"tictactoe"	958	0	9	Keine
"vote"	435	7	14	Einige

Tabelle 7: Verwendete Datensätze und deren Eigenschaften

Auf diesen Datensätzen wurden J48 und RandomJ48 Bäume diverser Komplexitäten erstellt und ihre Eigenschaften verglichen. Wird dabei von "Performanz" gesprochen, ist damit der voraussichtliche Anteil an vom Trainingsset unabhängigen Instanzen gemeint, welche der Baum korrekt klassifiziert. Zum Testen der Performanz wurde 10-fold-cross-validation verwendet. Dabei handelt es sich um eine Methode, bei welcher der Datensatz in zehn gleich große Abschnitte aufgeteilt wird, wovon bei zehn Durchgängen jeweils ein Abschnitt als Testset und die rechtlichen Abschnitte als Trainingsset fungieren. Da die Werte von RandomJ48-Bäumen aufgrund ihrer zufälligen Attributwahl auch auf dem gleichen Datensatz eine hohe Varianz aufweisen, wurden Experiment bezüglich RandomJ48 100-mal durchgeführt und die Durchschnittswerte betrachtet. Ergebnisse wurden auf 2 Nachkommstellen gerundet. Detailliertere Darstellungen der Experimente und die erhaltenen Datenreihen finden sich im Anhang.

Durchgeführt wurden diese Experimente in *Weka*[2]. Dabei handelt es sich um eine Open Source Data Mining Software, welche an der *University of Waikato* in Neuseeland entwickelt wurde. Unter *http://www.cs.waikato.ac.nz/ml/weka/* sind Software und weitere Informationen erhältlich.

[1] Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

4.1. Ohne Pruning

Zunächst werden Bäume der beiden Algorithmen verglichen, bei denen kein Pruning durchgeführt wurde.

4.1.1. Baumgröße

Da der Algorithmus RandomJ48 nicht der Heuristik unterliegt, einen möglichst kleinen Entscheidungsbaum zu erstellen, ist zu erwarten, dass RandomJ48-Bäume eine höhere Komplexität aufweisen als Bäume des J48 Algorithmus auf den gleichen Datensätzen. Um dies zu testen, wurden in folgender Tabelle die Komplexitäten der J48-Bäume und RandomJ48-Bäume jeweils ohne Pruning gegenübergestellt:

Datensatz	J48 ohne Pruning	RandomJ48 ohne Pruning	Vergleich
"anneal"	72	367,03	509,76 %
"horse-colic"	129	306,51	237,60 %
"flare"	192	424,50	221,09 %
"kr-vs-kp"	82	1470,26	1793 %
"mfeat"	241	1226,12	508,76 %
"segment"	101	1120,99	1109,89 %
"sonar"	35	155,73	444,94 %
"synthetic"	35	399,08	1140,23 %
"tictactoe"	208	551,97	265,37 %
"vote"	37	240,72	650,59 %
Durchschnitt			**688,12 %**

Tabelle 8: Baumgröße J48/RandomJ48 ohne Pruning

Im Schnitt ist ein RandomJ48-Baum auf den gewählten Datensätzen fast 7-mal größer als der J48-Baum auf den gleichen Daten. Dabei reicht die Bandbreite von etwas mehr als doppelter Größe (221,09% bei "flare") bis hin zu fast 18-facher Größe (1793% bei "kr-vs-kp"). Einen direkten Einfluss der Anzahl der Attribute und/oder deren Art auf die Baumgröße lässt sich auf Basis dieser Datensätze nicht herleiten.

4.1.2. Performanz

Da mehrere Durchführungen des J48-Algorithmus auf dem gleichen Datensatz immer einen identischen Entscheidungsbaum liefern, dies bei RandomJ48 aber nicht der Fall ist, gibt es beim Vergleich der Performanzen der beiden Algorithmen zwei Ansätze. Zunächst wurden der Größe der RandomJ48 Bäume keine Grenzen gesetzt und J48-Bäume mit "ausgewachsenen" RandomJ48 Bäumen verglichen:

Datensatz	J48 ohne Pruning	RandomJ48 ohne Pruning	Vergleich
"anneal"	98,44 %	94,89 %	96,39 %
"horse-colic"	82,34 %	74,09 %	89,98 %
"flare"	73,83 %	71,31 %	96,59 %
"kr-vs-kp"	99,41 %	83,80 %	84,30 %
"mfeat"	71,75 %	63,59 %	88,63 %
"segment"	97,06 %	80,72 %	83,17 %
"sonar"	69,71 %	62,00 %	88,94 %
"synthetic"	91,67 %	68,65 %	74,89 %
"tictactoe"	85,39 %	68,19 %	79,86 %
"vote"	96,32 %	90,57 %	95,07 %
Durchschnitt			**87,78 %**

Tabelle 9: Performanz J48/ungeschnittener RandomJ48 ohne Pruning

In diesem Fall erreicht RandomJ48 eine Performanz, welche ca. 88% der Performanz der gegenübergestellten J48 Bäume ausmacht.
Der zweite Ansatz ist ein Vergleich von Bäumen annähernd gleicher Komplexität. Hierfür wurden die RandomJ48-Bäume mit dem in Kapitel 3.2. vorgestellten Verfahren zurechtgeschnitten.

Datensatz	J48 ohne Pruning	RandomJ48 ohne Pruning	Vergleich
"anneal"	98,44 %	82,60 %	83,91 %
"horse-colic"	82,34 %	72,81 %	88,43 %
"flare"	73,83 %	58,63 %	79,41 %
"kr-vs-kp"	99,41 %	65,78 %	66,17 %
"mfeat"	71,75 %	62,37 %	86,93 %
"segment"	97,06 %	57,15 %	58,88 %
"sonar"	69,71 %	61,42 %	88,11 %
"synthetic"	91,67 %	50,11 %	54,66 %
"tictactoe"	85,39 %	68,36 %	80,06 %
"vote"	96,32 %	88,42 %	91,80 %
Durchschnitt			**77,84 %**

Tabelle 10: Performanz J48/geschnittener RandomJ48 ohne Pruning

Hierbei verschlechtert sich die relative Performanz von RandomJ48 gegenüber J48 auf ca. 78%.

4.2. Mit Pruning

Die folgenden Experimente behandeln Entscheidungsbäume, bei denen Pruning wie in Kapitel 2.7 beschrieben in den Entstehungsprozess miteinbezogen wurde.

4.2.1. Baumgröße

Auch bei den Varianten der Algorithmen mit Pruning werden zunächst die entstehenden Baumgrößen verglichen:

Datensatz	J48 mit Pruning	RandomJ48 mit Pruning	Vergleich
"anneal"	47	69,59	148,06 %
"horse-colic"	6	10,02	167 %
"flare"	50	40,10	80,2 %
"kr-vs-kp"	59	69,19	117,27 %
"mfeat"	137	141,75	103,47 %
"segment"	77	186,55	242,27 %
"sonar"	35	21,75	62,14 %
"synthetic"	35	72,99	208,54 %
"tictactoe"	142	73,42	51,70 %
"vote"	11	7,62	69,27 %
Durchschnitt			**124,99 %**

Tabelle 11: Baumgröße J48/RandomJ48 mit Pruning

Zwar scheinen die RandomJ48 Bäume mit im Schnitt 124,99% der Komplexität der J48 Bäume etwas an Größe zuzunehmen, allerdings nimmt sie in vier Fällen ("flare", "sonar", "tictactoe" und "vote") ab, und die relativ geringe Knotenanzahl lässt auf eine hohe Varianz der Vergleichswerte schließen. Von einem signifikanten Wachstum kann also, zumindest auf Basis dieser Daten, nicht gesprochen werden.

4.2.2. Performanz

Die Performanz der Algorithmen wird, analog zum Vergleich der Varianten ohne Pruning, zunächst ohne Zurechtschneiden der RandomJ48-Bäume betrachtet:

Datensatz	J48 mit Pruning	RandomJ48 mit Pruning	Verhältnis
"anneal"	98,44 %	95,88 %	97,40 %
"horse-colic"	85,33 %	76,41 %	89,55 %
"flare"	74,48 %	74,71 %	100,31 %
"kr-vs-kp"	99,44 %	94,20 %	94,73 %
"mfeat"	72,05 %	69,91 %	97,03 %
"segment"	96,93 %	84,03 %	86,69 %
"sonar"	71,15 %	65,00 %	91,36 %
"synthetic"	91,67 %	72,65 %	79,25 %
"tictactoe"	84,55 %	76,16 %	91,08 %
"vote"	96,32 %	94,37 %	97,98 %
Durchschnitt			**92,54 %**

Tabelle 12: Performanz J48/ungeschnittener RandomJ48 mit Pruning

Hierbei erreicht RandomJ48 ~93% der Performanz von J48. Ein Fall („flare“) fällt dabei aus dem Rahmen, da bei ihm eine leichte Erhöhung der Performanz zu erkennen ist.

Bei Bäumen annähernd gleicher Komplexität entstehen folgende Werte bezüglich ihrer Performanz:

Datensatz	J48 mit Pruning	RandomJ48 mit Pruning	Verhältnis
"anneal"	98,44 %	82,40 %	83,71 %
"horse-colic"	85,33 %	72,94 %	85,48 %
"flare"	74,48 %	61,27 %	82,26 %
"kr-vs-kp"	99,44 %	67,14 %	67,52 %
"mfeat"	72,05 %	63,96 %	88,78 %
"segment"	96,93 %	58,03 %	59,87 %
"sonar"	71,15 %	62,05 %	87,21 %
"synthetic"	91,67 %	50,98 %	55,61 %
"tictactoe"	84,55 %	71,91 %	85,05 %
"vote"	96,32 %	89,98 %	93,42 %
Durchschnitt			**78,89 %**

Tabelle 13: Performanz J48/geschnittener RandomJ48 mit Pruning

Die Performanz von RandomJ48 beträgt hier ~79% des Wertes von J48.

4.3. Auswirkung des Prunings auf die Baumgröße

Da Pruning in der Regel die Performanz eines Entscheidungsbaum verbessert, kann das Verhältnis der Größe des Baumes vor und nach dem Pruning-Vorgang als Qualitätsmerkmal des zugrunde liegenden Algorithmus angesehen werden: Je weniger der Baum durch Pruning an Komplexität verliert, desto besser war der ursprünglich entstandene Baum.
Das Verhältnis der Größe der J48-Bäume mit und ohne Pruning ist folgender Tabelle zu entnehmen:

Datensatz	J48 ohne Pruning	J48 mit Pruning	Verhältnis
"anneal"	72	47	65,78 %
"horse-colic"	129	6	4,65 %
"flare"	192	50	26,04 %
"kr-vs-kp"	82	59	71,95 %
"mfeat"	241	137	56,85 %
"segment"	101	77	76,24 %
"sonar"	35	35	100 %
"synthetic"	35	35	100 %
"tictactoe"	208	142	68,27 %
"vote"	37	11	29,73 %
Durchschnitt			**59,95 %**

Tabelle 14: Baumgröße J48 vor und nach Pruning

Bei RandomJ48 ergeben sich folgende Werte:

Datensatz	RandomJ48 ohne Pruning	RandomJ48 mit Pruning	Verhältnis
"anneal"	72	21,21	29,46 %
"horse-colic"	129	9,90	7,67 %
"flare"	192	35,33	18,40 %
"kr-vs-kp"	82	21,70	26,46 %
"mfeat"	241	54,52	22,62 %
"segment"	101	40,32	39,92 %
"sonar"	35	8,18	23,37 %
"synthetic"	35	18,23	52,09 %
"tictactoe"	208	37,90	18,22 %
"vote"	37	7,03	19,00 %
Durchschnitt			**25,72 %**

Tabelle 15: Baumgröße RandomJ49 vor und nach Pruning

Während also bei J48 Bäume durch Pruning auf ~60% ihrer Größe schrumpfen, erreichen RandomJ48 Bäume durch Pruning lediglich ~26% der Größe ihrer Ausgangsbäume.

4.4. Auswirkung des Prunings auf die Performanz

Auch die erreichte Performanzsteigerung durch Pruning kann als Qualitätsmerkmal eines Algorithmus dienen: Je weniger Performanz ein Baum durch Pruning gewinnt, desto besser ist der Algorithmus, der ihn erstellt hat. Hier wird zunächst J48 untersucht:

Datensatz	J48 ohne Pruning	J48 mit Pruning	Verhältnis
"anneal"	98,44 %	98,44 %	100 %
"horse-colic"	82,34 %	85,33 %	103,63 %
"flare"	73,83 %	74,48 %	100,88 %
"kr-vs-kp"	99,41 %	99,44 %	100,03 %
"mfeat"	71,75 %	72,05 %	100,42 %
"segment"	97,06 %	96,93 %	99,87 %
"sonar"	69,71 %	71,15 %	102,07 %
"synthetic"	91,67 %	91,67 %	100 %
"tictactoe"	85,39 %	84,55 %	99,02 %
"vote"	96,32 %	96,32 %	100 %
Durchschnitt			**100,59 %**

Tabelle 16: Performanz J48 vor und nach Pruning

Die in den Experimenten erstellten J48-Bäume erhielten durch Pruning eine Performanzsteigerung um ~0,6%. Bei RandomJ48 sieht es folgendermaßen aus:

Datensatz	RandomJ48 ohne Pruning	RandomJ48 mit Pruning	Verhältnis
"anneal"	82,60 %	95,88 %	116,08 %
"horse-colic"	72,81 %	76,41 %	104,94 %
"flare"	58,63 %	74,71 %	127,43 %
"kr-vs-kp"	65,78 %	94,20 %	143,20 %
"mfeat"	62,37 %	69,91 %	112,09 %
"segment"	57,15 %	84,03 %	147,03 %
"sonar"	61,42 %	65,00 %	105,83 %
"synthetic"	50,11 %	72,65 %	144,98 %
"tictactoe"	68,36 %	76,16 %	111,41 %
"vote"	88,42 %	94,37 %	106,73 %
Durchschnitt			**121,97 %**

Tabelle 17: Performanz RandomJ48 vor und nach Pruning

Bei RandomJ48 steigt die Performanz der Bäume durch Pruning auf ~122 % ihres Ausgangswertes.

4.5. Zusammenfassung

Die Ergebnisse der Analyse der Performanzen der verglichenen Algorithmen sind in folgender Tabelle zusammengefasst:

	Zurechtgeschnitten	Nicht Zurechtgeschnitten
Ohne Pruning	**77,84 %**	**87,78 %**
Mit Pruning	**78,89 %**	**92,54 %**

Tabelle 18: Verhältnis der Performanzen RandomJ48 zu J48

Wie erwartet schneidet J48 bezüglich der Performanz bei gleicher Baumgröße besser ab. RandomJ48 erreicht hier Werte von ~78% (ohne Pruning) bzw. ~79% (mit Pruning) der Vergleichswerte. Der Unterschied kann um ~10 Prozentpunkte (ohne Pruning) bzw. ~14 Prozentpunkte (mit Pruning) verringert werden, indem man der Größe der RandomJ48-Bäume keine Grenzen setzt. Durch Pruning verringert sich der Unterschied ebenfalls um ~1 Prozentpunkt (bei geschnittenen Bäumen) bzw. ~5 Prozentpunkten (bei ungeschnittenen Bäumen), was darauf zurückzuführen ist, dass Pruning auf RandomJ48 größere Auswirkungen hat als auf J48. Dies erkennt man ebenfalls an den Baumgrößen: Bei J48 verringert Pruning die Komplexität auf ~60% des Ausgangswertes, bei RandomJ48 sind es ~26%.

Die Effektivität des Algorithmus J48 beim Einhalten der von *Occam's Razor* motivierten Heuristik, möglichst kleine Entscheidungsbäume zu erhalten, ist beim Vergleich klar erkennbar: Der Algorithmus RandomJ48, der dieser Heuristik keine Beachtung schenkt, produziert in seiner Variante ohne Pruning im Schnitt 7-mal größere Bäume.

Die Performanzen der RandomJ48 Bäume verbessern sich sowohl in der Variante ohne als auch in der Variante mit Pruning bei einer Erhöhung der Anzahl der Knoten zunächst stark. Mit weiterer Hinzunahme von Knoten wird das Wachstum geringer und die Performanzen nähern sich asymptotisch ihrem Maximalwert an. Overfitting und dessen Konsequenzen sind nicht zu beobachten.[1]

[1] Siehe dazu die Graphen zur Performance bei steigender Komplexität im Anhang.

5. Ausblick

In dieser Arbeit wurde RandomJ48 auf Datensätzen mit unterschiedlichen Anzahlen an numerischen und nominellen Attributen getestet, wobei jeweils der Durchschnitt der erhaltenen Werte betrachtet wurde. Es bleibt zu überprüfen, wie sehr das Verhältnis zwischen den verschiedenen Attributarten die Performanz und die Baumgröße beeinflusst. Da RandomJ48 bei numerischen Attributen einen zufälligen Splitpoint wählt, was bei nominellen Attributen nicht nötig ist, ist zu erwarten, dass RandomJ48 bei Datensätzen mit einem hohen Anteil an numerischen Attributen gegenüber J48 weiter an Performanz einbüßt. Die Anzahl an Datensätzen, die in dieser Arbeit verwendet wurde, ist zu klein, um diese These zu belegen.

Da auf den verwendeten Datensätzen RandomJ48 keine Anzeichen von Overfitting aufzeigt, sollte überprüft werden, ob dieses Verhalten auf weiteren Datensätzen bestätigt werden kann. Insbesondere sollten dabei Experimente auf Datensätzen mit einem Anteil an fehlerhaft klassifizierten Instanzen durchgeführt werden, da diese in der Regel häufiger zu Overfitting führen als Datensätze aus komplett korrekt klassifizierten Instanzen, wie sie in dieser Arbeit ausschließlich verwendet wurden.

Weiterhin wären Vergleiche zur benötigten Rechenzeit von RandomJ48 mit J48 oder anderen Algorithmen interessant. Da RandomJ48 keine Entropie-Berechnungen durchführt, sollte er gegenüber J48 die Attributswahl schneller ausführen. Allerdings muss er dafür in seiner ungeschnittenen Variante mehr Knoten erstellen und in seiner geschnittenen mehr Rechenzeit zur Entfernung von Knoten aufwenden. Das Verhältnis zwischen zusätzlicher Rechenzeit und zusätzlicher Performanz kann ein wichtiger Faktor sein bei der Bewertung von Algorithmen, die in der Praxis eingesetzt werden. Insbesondere sei hierbei auf eine Publikation verwiesen, laut deren Angaben bei den meisten Problemen ein Zusammenschluss aus zehn zufällig erstellten Entscheidungsbäumen einem C4.5 Baum überlegen ist, was sowohl Performanz als auch Rechenaufwand angeht [WEI03].

Schließlich können Vergleiche zwischen RandomJ48 und weiteren Algorithmen durchgeführt werden, so dass RandomJ48 die Rolle eines Benchmark-Algorithmus einnimmt. Um eine Vergleichbarkeit zu gewährleisten, sollte dabei darauf geachtet werden, dass Konzepte des zu vergleichenden Algorithmus, wie das Verhalten bei fehlenden Attributwerten oder ein mögliches alternative Pruning, in den Aufbau von RandomJ48 übernommen werden, sodass sich die Algorithmen nur in ihrer Attributwahl unterscheiden und keine weiteren Unterschiede den Vergleich beeinflussen.

Literaturverzeichnis

[ALP08] Alpaydin E. (2008). *Maschinelles Lernen.* Übersetzt von Simone Linke. Oldenbourg. 183 - 195.

[MIT97] Mitchell T. (1997). *Machine Learning*. International Edition. McGraw Hill. 52 - 80.

[QUI93] Quinlan J.R.(1993). *C4.5: Programs for Machine Learning.* Morgan Kaufman. vii, 17-43

[WEI03] Fan W., Wang H., Yu P.S. und Ma S. (2003). *Is random Model better? On its accuracy and efficiency.* 3rd IEEE International Conference on Data Mining (ICDM 2003), November 19-22, Melbourne.

[WIT05] Witten I.H. & Frank E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Second Edition. Morgan Kaufmann. 187 -194.

Anhang

Im Folgenden werden die erhaltenen Datenreihen der durchgeführten Experimente dargestellt. Für jeden Datensatz wird dieser zunächst vorgestellt, die Anzahl seiner Instanzen, Attribute sowie deren Art aufgelistet, und festgestellt, ob er fehlende Attributwerte enthält. Daraufhin wird J48 mit und ohne Pruning ausgeführt. Dem folgen Ausführungen von RandomJ48 mit und ohne Pruning bei jeweils steigender Komplexität. Schließlich folgen zu Vergleichszwecken noch eine Durchführung von RandomJ48 mit der gleichen Komplexität wie der bei J48 entstandenen, sowie eine ohne Beschränkung der Komplexität. Schließlich werden zu jedem Datensatz noch zwei Graphen dargestellt. Der erste zeigt das Verhalten der Performanz von RandomJ48-Bäumen mit und ohne Pruning bei steigender Komplexität, der zweite beschreibt die Komplexität der entstehenden RandomJ48-Bäume nach dem Pruning bei steigender Baumgröße.

Datensatz „anneal“
http://repository.seasr.org/Datasets/UCI/arff/anneal.arff
898 Instanzen, 39 Attribute (6 numerisch + 32 nominell + Klassenattribut)
Viele fehlende Attributwerte

Unpruned J48:

Anzahl Blätter	**53**
Komplexität	**72**
Tiefe	**13**
Korrekt klassifizierte Instanzen	**98,441%**

Pruned J48:

Anzahl Blätter	**35**
Komplexität	**47**
Tiefe	**8**
Korrekt klassifizierte Instanzen	**98,441 %**

100-mal jeweils 10-folds cross-validation Unpruned RandomJ48 bei steigender Komplexität

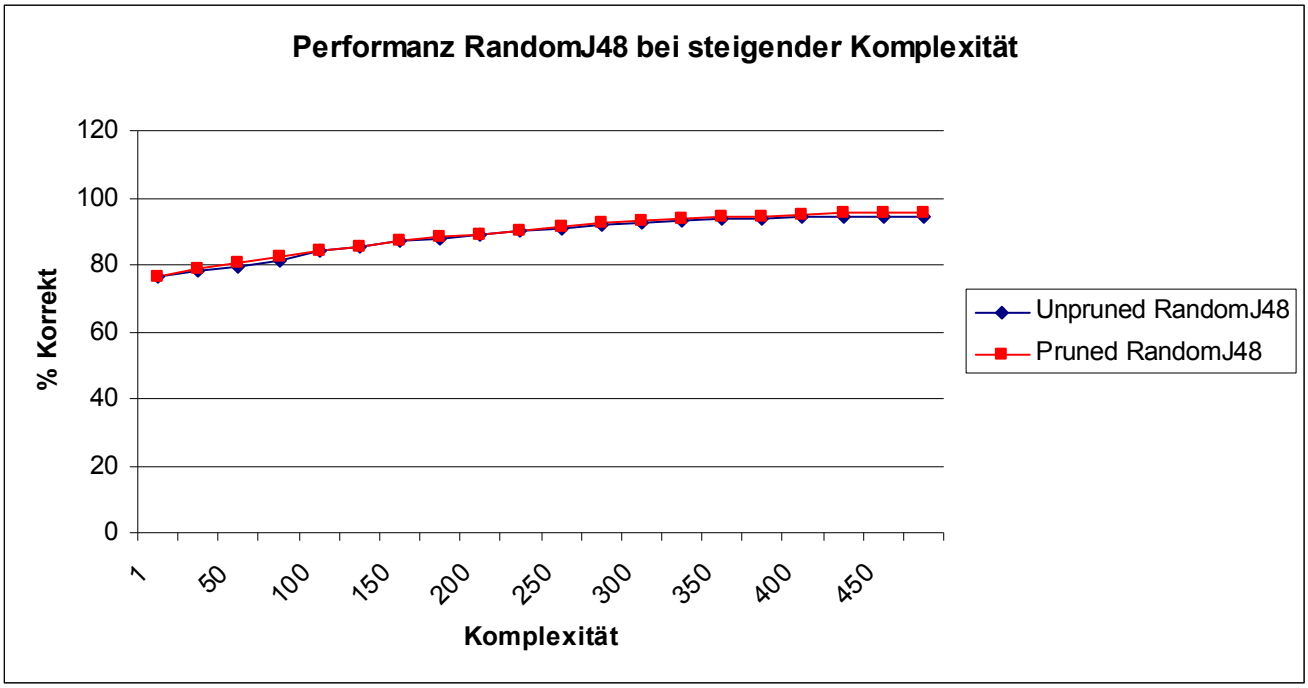
Kompl.	**1**	**25**	**50**	**75**	**100**	**125**	**150**	**175**	**200**	**225**
% korrekt	76,17	78,07	79,51	80,94	84,02	85,47	86,89	87,80	88,87	90,02
Kompl.	**250**	**275**	**300**	**325**	**350**	**375**	**400**	**425**	**450**	**475**
% korrekt	90,88	91,87	92,57	93,14	93,57	93,93	94,21	94,15	94,34	94,50

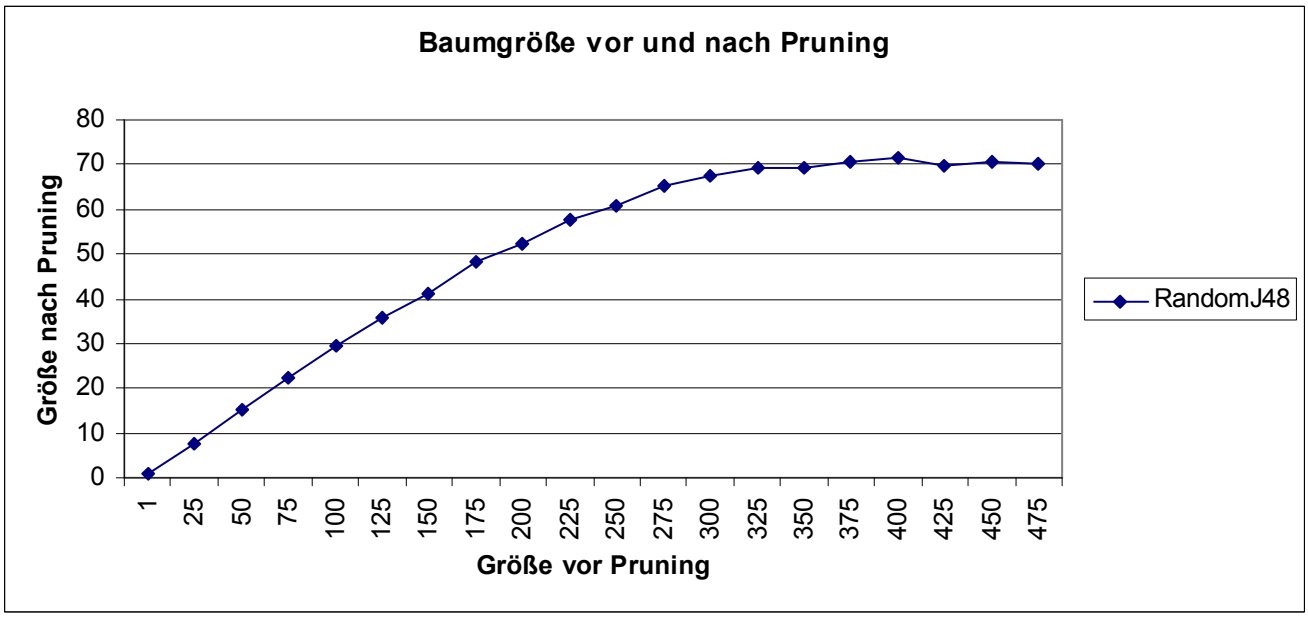
Kompl.	72	max
# Knoten		367,03
% korrekt	82,60	94,89

100-mal jeweils 10-folds cross-validation Pruned RandomJ48 bei steigender Komplexität

Kompl.	**1**	**25**	**50**	**75**	**100**	**125**	**150**	**175**	**200**	**225**
# Knoten	1	7,75	15,34	22,45	29,69	35,92	41,25	48,07	52,31	57,44
% korrekt	76,17	78,82	80,66	82,54	84,29	85,47	86,96	88,30	89,25	90,31
Kompl.	**250**	**275**	**300**	**325**	**350**	**375**	**400**	**425**	**450**	**475**
# Knoten	60,97	65,16	67,69	69,23	69,06	70,79	71,29	69,84	70,80	70,08
% korrekt	91,46	92,37	92,86	93,61	94,43	94,56	95,02	95,32	95,58	95,66

Kompl.	72	max
# Knoten	21,21	69,59
% korrekt	82,40	95,88


Performanz RandomJ48 bei steigender Komplexität
% Korrekt
120
100
80
60
40
20
0
1
50
100
150
200
250
300
350
400
450
Komplexität
Unpruned RandomJ48
Pruned RandomJ48


Baumgröße vor und nach Pruning
Größe nach Pruning
80
70
60
50
40
30
20
10
0
1
25
50
75
100
125
150
175
200
225
250
275
300
325
350
375
400
425
450
475
Größe vor Pruning
RandomJ48

Datensatz „horse-colic“
http://repository.seasr.org/Datasets/UCI/arff/colic.arff
368 Instanzen, 22 Attribute (7 numerisch + 14 nominell + Klassenattribut)
Einige fehlende Attributwerte

Unpruned J48:

Anzahl Blätter	**95**
Komplexität	**129**
Tiefe	**7**
Korrekt klassifizierte Instanzen	**82,337%**

Pruned J48:

Anzahl Blätter	**4**
Komplexität	**6**
Tiefe	**3**
Korrekt klassifizierte Instanzen	**85,3261 %**

100-mal jeweils 10-folds cross-validation Unpruned RandomJ48 bei steigender Komplexität

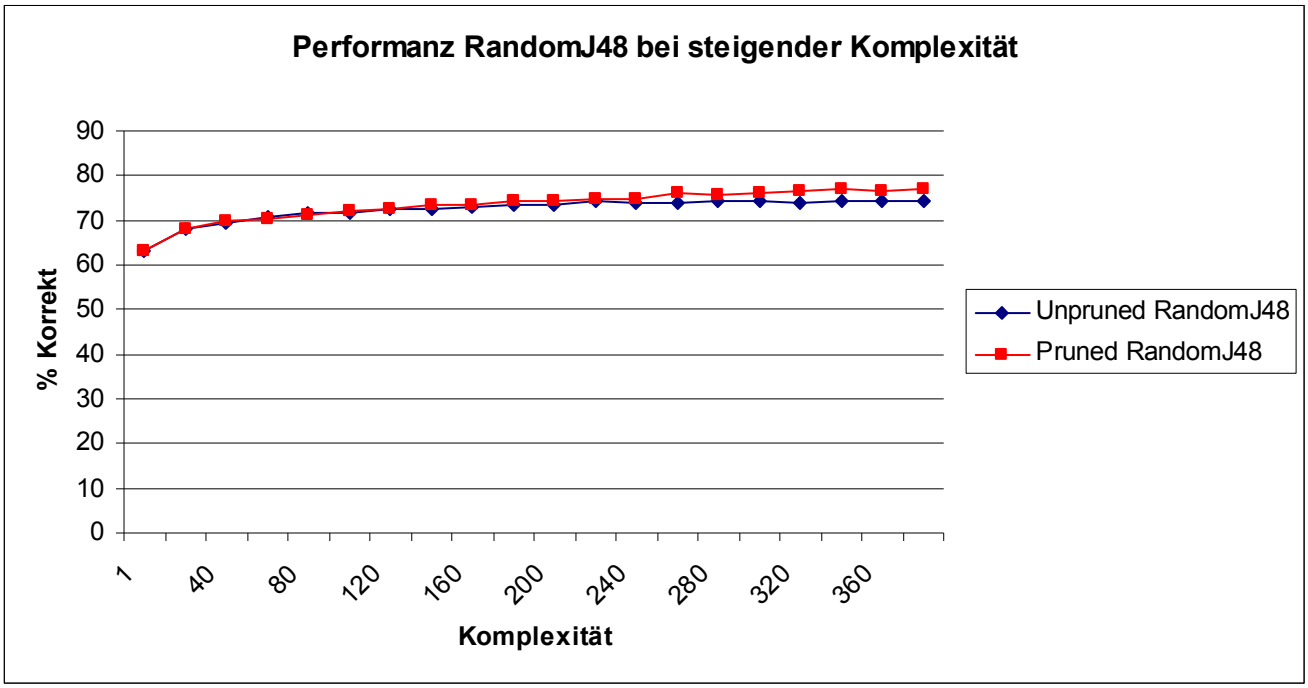
Kompl.	**1**	**20**	**40**	**60**	**80**	**100**	**120**	**140**	**160**	**180**
% korrekt	63,05	68,17	69,55	70,92	71,59	71,76	72,67	72,59	73,20	73,46
Kompl.	**200**	**220**	**240**	**260**	**280**	**300**	**320**	**340**	**360**	**380**
% korrekt	73,56	74,13	73,82	74,05	74,28	74,50	73,93	74,39	74,24	74,51

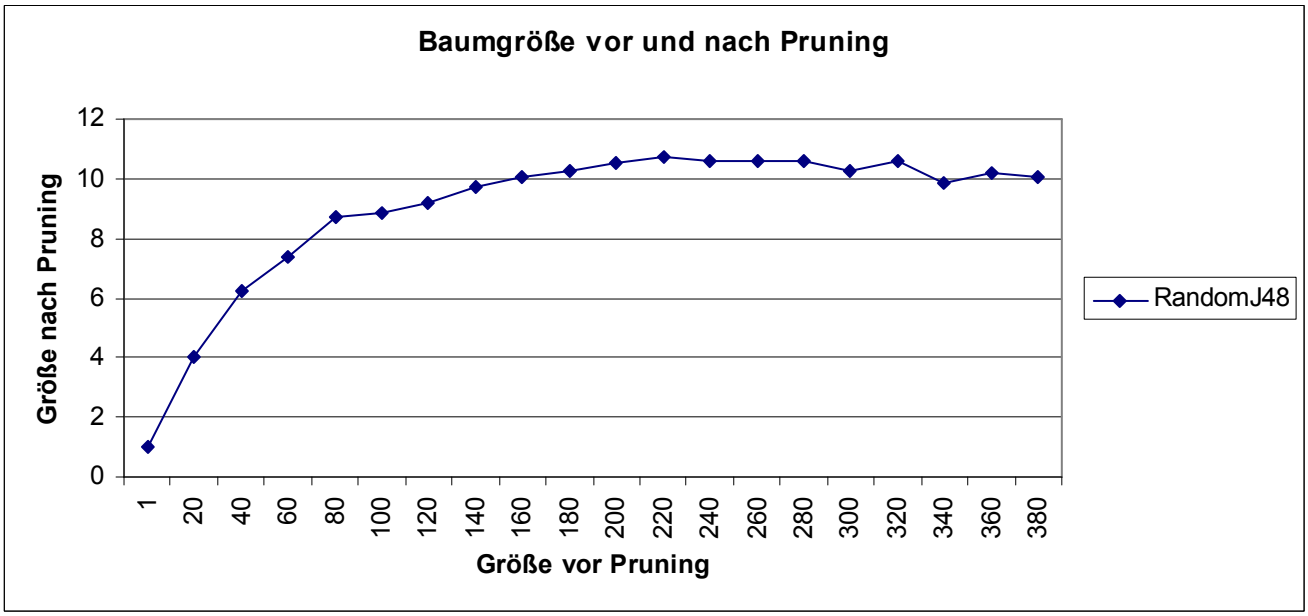
Kompl.	129	max
# Knoten		306,51
% korrekt	72,81	74,09

100-mal jeweils 10-folds cross-validation Pruned RandomJ48 bei steigender Komplexität

Kompl.	**1**	**20**	**40**	**60**	**80**	**100**	**120**	**140**	**160**	**180**
# Knoten	1	4,04	6,25	7,40	8,73	8,83	9,19	9,69	10,07	10,23
% korrekt	63,05	68,08	69,74	70,32	71,01	71,88	72,61	73,34	73,40	74,46
Kompl.	**200**	**220**	**240**	**260**	**280**	**300**	**320**	**340**	**360**	**380**
# Knoten	10,54	10,75	10,57	10,61	10,57	10,27	10,59	9,86	10,22	10,06
% korrekt	74,32	74,66	74,95	75,95	75,50	76,33	76,69	76,9	76,59	76,86

Kompl.	129	max
# Knoten	9,90	10,02
% korrekt	72,94	76,41


Performanz RandomJ48 bei steigender Komplexität
% Korrekt
90
80
70
60
50
40
30
20
10
0
1
40
80
120
160
200
240
280
320
360
Komplexität
Unpruned RandomJ48
Pruned RandomJ48


Baumgröße vor und nach Pruning
Größe nach Pruning
12
10
8
6
4
2
0
1
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
360
380
Größe vor Pruning
RandomJ48

Datensatz „flare“
http://repository.seasr.org/Datasets/UCI/arff/solar-flare_2.arff
1066 Instanzen, 13 Attribute (12 nominell + Klassenattribut)
Keine fehlenden Attributwerte

Unpruned J48:

Anzahl Blätter	**145**
Komplexität	**192**
Tiefe	**8**
Korrekt klassifizierte Instanzen	**73,8274%**

Pruned J48:

Anzahl Blätter	**38**
Komplexität	**50**
Tiefe	**7**
Korrekt klassifizierte Instanzen	**74,4841 %**

100-mal jeweils 10-folds cross-validation Unpruned RandomJ48 bei steigender Komplexität

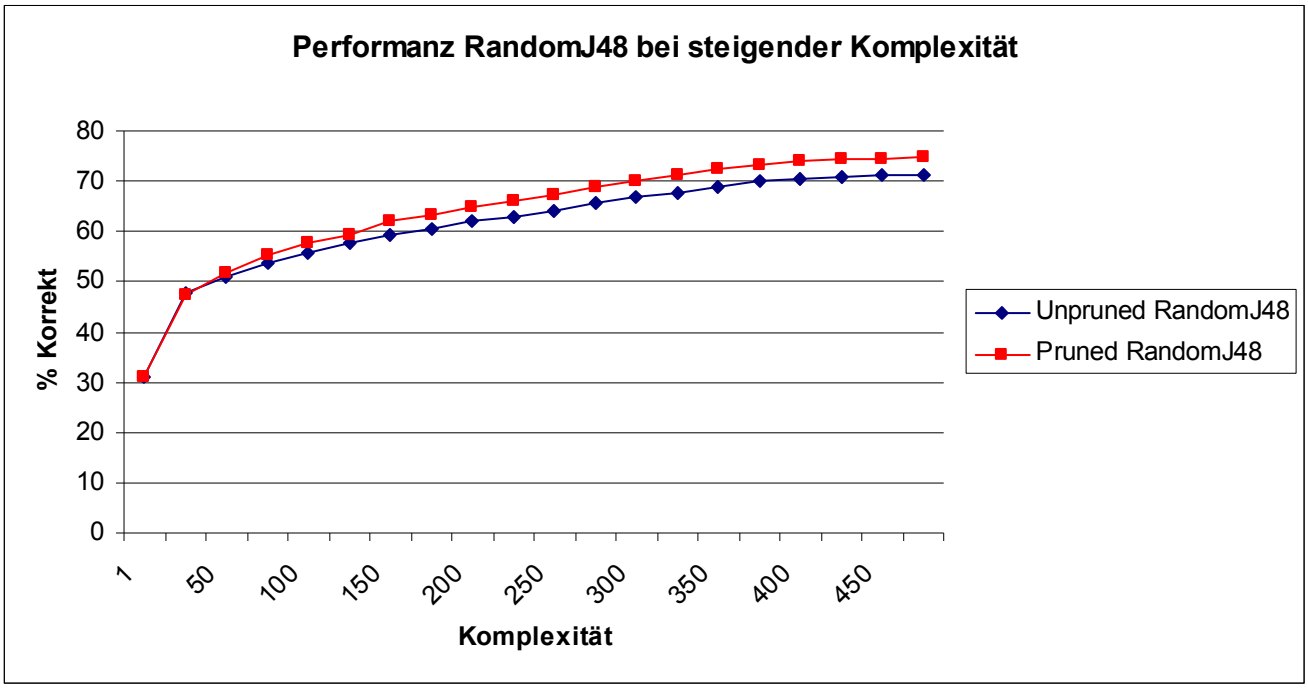
Kompl.	**1**	**25**	**50**	**75**	**100**	**125**	**150**	**175**	**200**	**225**
% korrekt	31,05	47,73	50,99	53,81	55,68	57,77	59,40	60,33	61,97	63,00
Kompl.	**250**	**275**	**300**	**325**	**350**	**375**	**400**	**425**	**450**	**475**
% korrekt	64,12	65,7	66,71	67,74	68,78	69,86	70,61	71,03	71,29	71,42

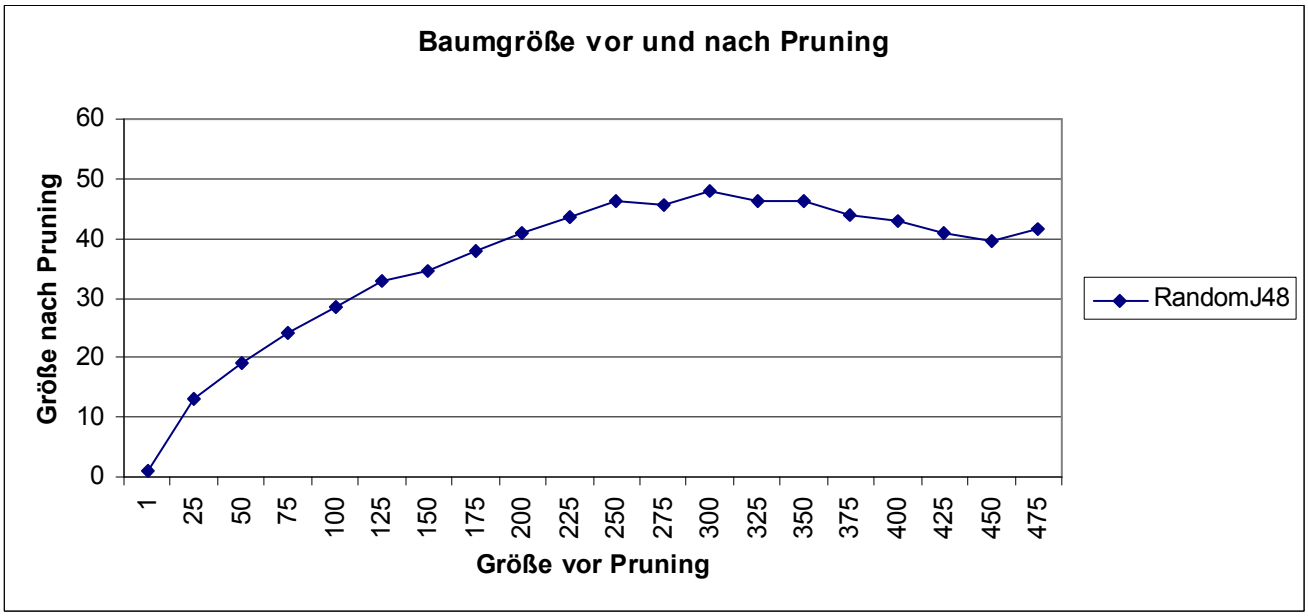
Kompl.	145	max
# Knoten		424,50
% korrekt	58,63	71,31

100-mal jeweils 10-folds cross-validation Pruned RandomJ48 bei steigender Komplexität

Kompl.	**1**	**25**	**50**	**75**	**100**	**125**	**150**	**175**	**200**	**225**
# Knoten	1	13,05	19,06	24,25	28,34	32,86	34,42	37,91	40,77	43,46
% korrekt	31,05	47,53	51,80	55,13	57,85	59,48	61,94	63,45	64,87	66,10
Kompl.	**250**	**275**	**300**	**325**	**350**	**375**	**400**	**425**	**450**	**475**
# Knoten	46,12	45,62	47,77	46,21	46,13	43,81	43,03	40,85	39,56	41,53
% korrekt	67,23	68,75	69,90	71,23	72,35	73,19	73,96	74,29	74,55	74,64

Kompl.	145	max
# Knoten	35,33	40,10
% korrekt	61,27	74,71


Performanz RandomJ48 bei steigender Komplexität
% Korrekt
80
70
60
50
40
30
20
10
0
1
50
100
150
200
250
300
350
400
450
Komplexität
Unpruned RandomJ48
Pruned RandomJ48


Baumgröße vor und nach Pruning
Größe nach Pruning
60
50
40
30
20
10
0
1
25
50
75
100
125
150
175
200
225
250
275
300
325
350
375
400
425
450
475
Größe vor Pruning
RandomJ48

Datensatz „kr-vs-kp“
http://repository.seasr.org/Datasets/UCI/arff/kr-vs-kp.arff
3196 Instanzen, 36 Attribute (35 nominell + Klassenattribut)
Keine fehlenden Attributwerte

Unpruned J48:

Anzahl Blätter	**43**
Komplexität	**82**
Tiefe	**14**
Korrekt klassifizierte Instanzen	**99,4055 %**

Pruned J48:

Anzahl Blätter	**31**
Komplexität	**59**
Tiefe	**14**
Korrekt klassifizierte Instanzen	**99,4368 %**

100-mal jeweils 10-folds cross-validation Unpruned RandomJ48 bei steigender Komplexität

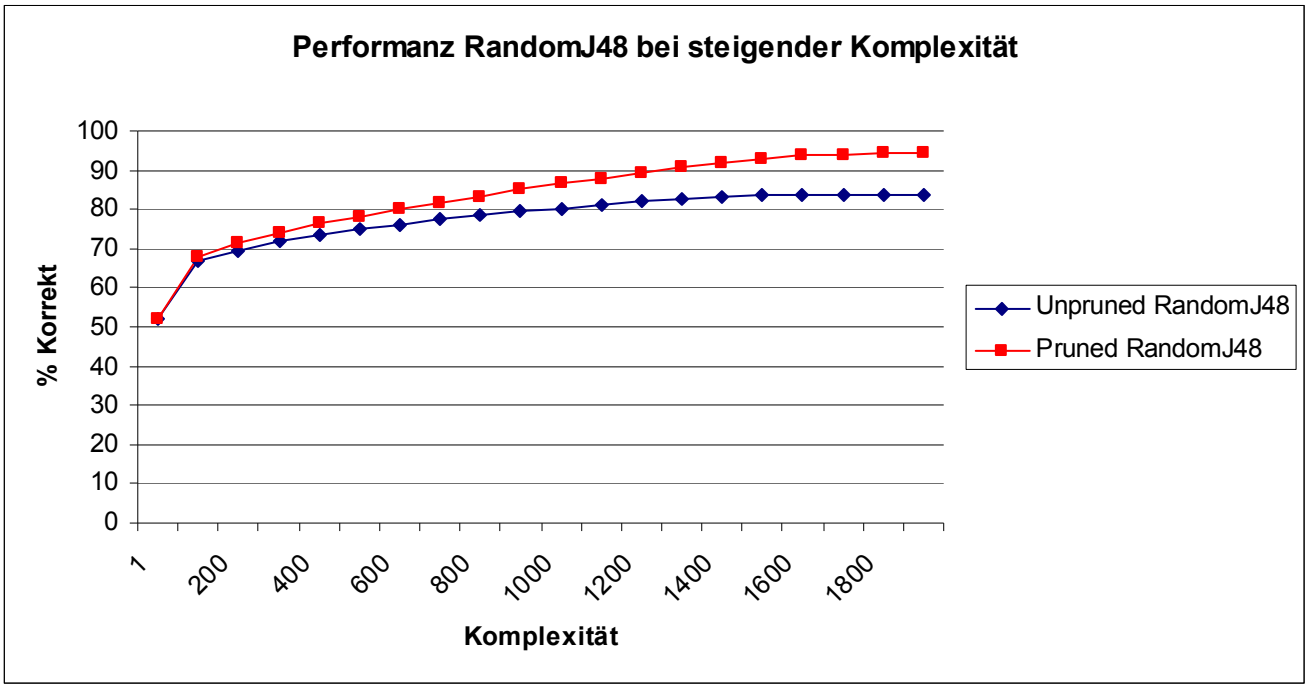
Kompl.	**1**	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**
% korrekt	52,22	66,85	69,59	71,77	73,47	74,80	76,05	77,31	78,50	79,55
Kompl.	**1000**	**1100**	**1200**	**1300**	**1400**	**1500**	**1600**	**1700**	**1800**	**1900**
% korrekt	79,96	80,89	81,95	82,58	83,11	83,50	83,85	83,91	83,76	83,77

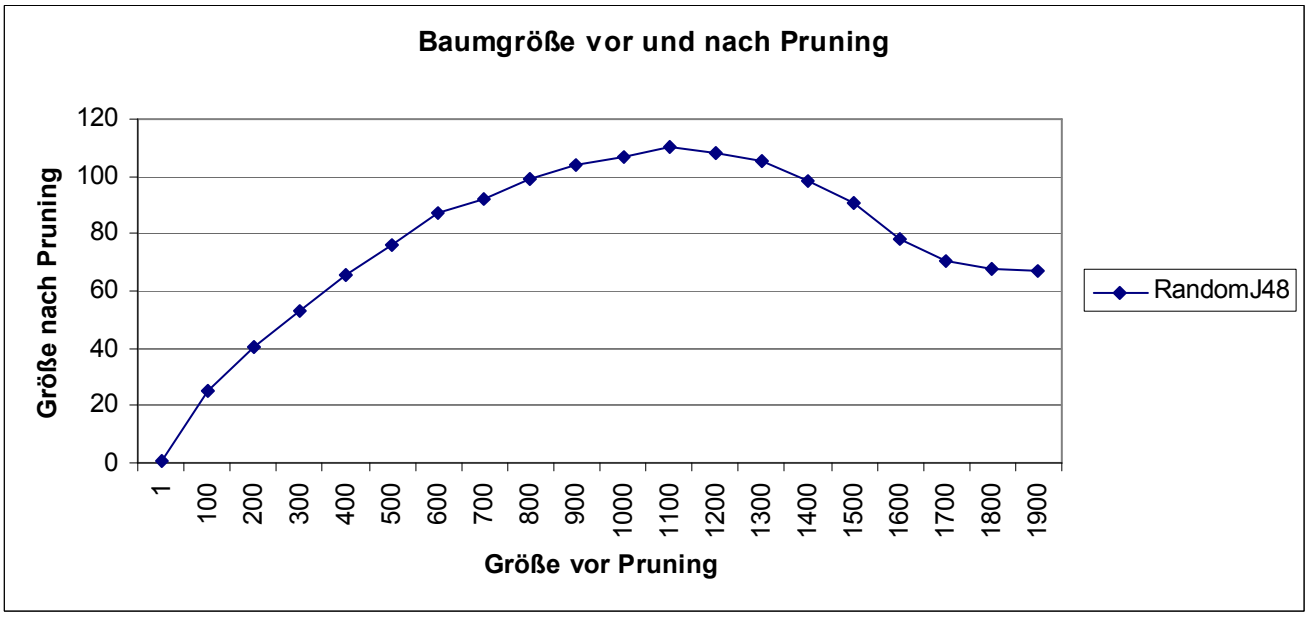
Kompl.	82	max
# Knoten		1470,26
% korrekt	65,78	83,80

100-mal jeweils 10-folds cross-validation Pruned RandomJ48 bei steigender Komplexität

Kompl.	**1**	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**
# Knoten	1	24,98	40,33	53,06	65,35	76,38	87,07	92,21	99	104,28
% korrekt	52,22	68,11	71,64	74,22	76,54	78,20	79,96	81,60	83,35	85,02
Kompl.	**1000**	**1100**	**1200**	**1300**	**1400**	**1500**	**1600**	**1700**	**1800**	**1900**
# Knoten	106,51	109,97	108,23	105,05	98,32	90,77	78,31	70,56	67,35	67,32
% korrekt	86,89	87,78	89,22	90,63	91,89	92,71	93,72	93,98	94,18	94,18

Kompl.	82	max
# Knoten	21,70	69,19
% korrekt	67,14	94,20


Performanz RandomJ48 bei steigender Komplexität
% Korrekt
100
90
80
70
60
50
40
30
20
10
0
1
200
400
600
800
1000
1200
1400
1600
1800
Komplexität
Unpruned RandomJ48
Pruned RandomJ48


Baumgröße vor und nach Pruning
Größe nach Pruning
120
100
80
60
40
20
0
1
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1700
1800
1900
Größe vor Pruning
RandomJ48

Datensatz „mfeat“
http://repository.seasr.org/Datasets/UCI/arff/mfeat-morphological.arff
2000 Instanzen, 6 Attribute (5 numerisch + Klassenattribut)
Keine fehlenden Attributwerte

Unpruned J48:

Anzahl Blätter	**121**
Komplexität	**241**
Tiefe	**15**
Korrekt klassifizierte Instanzen	**71,75 %**

Pruned J48:

Anzahl Blätter	**69**
Komplexität	**137**
Tiefe	**15**
Korrekt klassifizierte Instanzen	**72,05 %**

100-mal jeweils 10-folds cross-validation Unpruned RandomJ48 bei steigender Komplexität

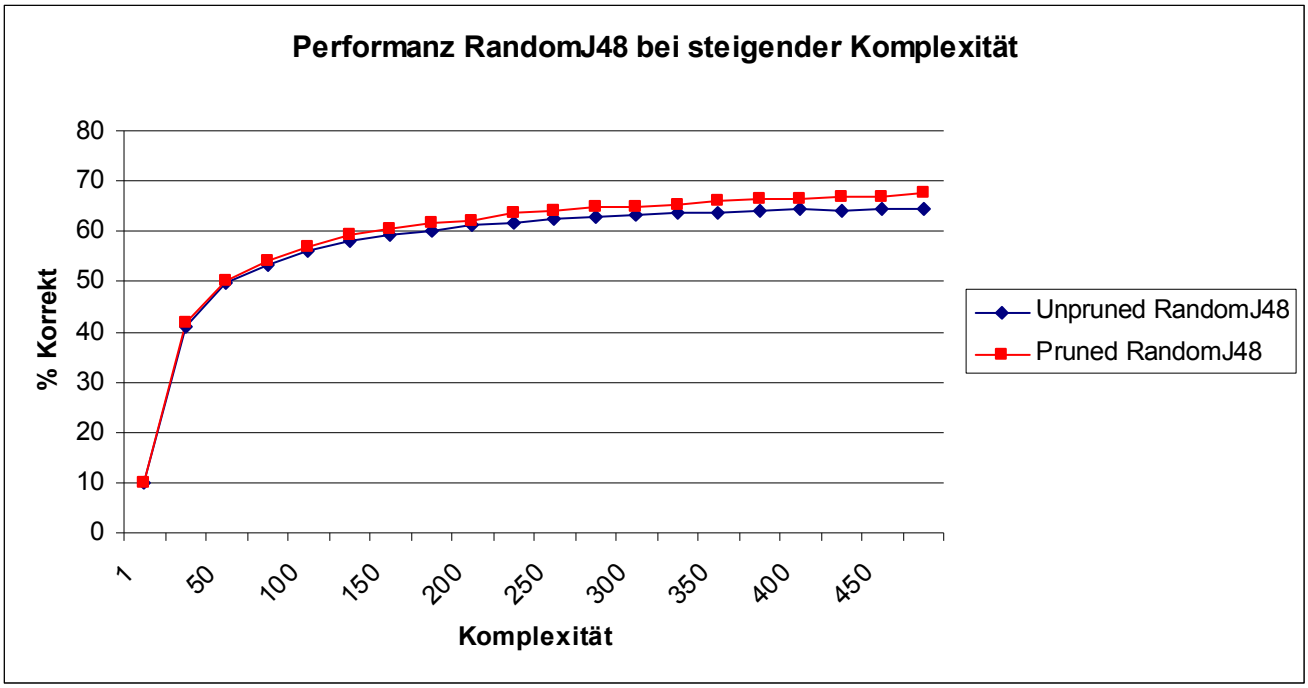
Kompl.	**1**	**25**	**50**	**75**	**100**	**125**	**150**	**175**	**200**	**225**
% korrekt	10	41,14	49,56	53,30	55,98	58,22	59,45	60,14	61,10	61,79
Kompl.	**250**	**275**	**300**	**325**	**350**	**375**	**400**	**425**	**450**	**475**
% korrekt	62,47	63,04	63,20	63,49	63,86	64,24	64,29	64,23	64,57	64,64

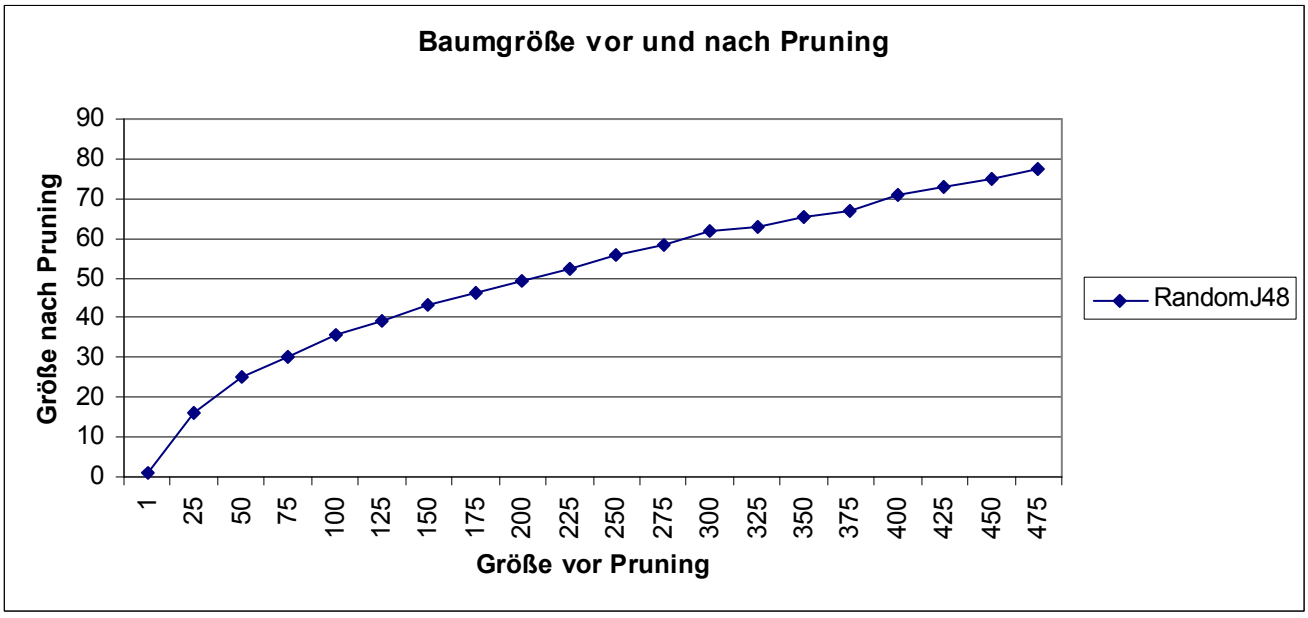
Kompl.	241	max
# Knoten		1226,12
% korrekt	62,37	63,59

100-mal jeweils 10-folds cross-validation Pruned RandomJ48 bei steigender Komplexität

Kompl.	**1**	**25**	**50**	**75**	**100**	**125**	**150**	**175**	**200**	**225**
# Knoten	1	16,17	24,92	30,31	35,56	39,06	43,24	46,17	49,32	52,46
% korrekt	10	41,99	50,10	54,25	56,94	59,11	60,62	61,80	62,22	63,54
Kompl.	**250**	**275**	**300**	**325**	**350**	**375**	**400**	**425**	**450**	**475**
# Knoten	55,66	58,48	61,63	62,75	65,49	66,87	70,68	72,83	74,68	77,31
% korrekt	64,21	64,74	65,01	65,46	65,96	66,40	66,52	66,97	66,98	67,58

Kompl.	241	max
# Knoten	54,52	141,75
% korrekt	63,96	69,91


Performanz RandomJ48 bei steigender Komplexität
% Korrekt
80
70
60
50
40
30
20
10
0
1
50
100
150
200
250
300
350
400
450
Komplexität
Unpruned RandomJ48
Pruned RandomJ48


Baumgröße vor und nach Pruning
Größe nach Pruning
90
80
70
60
50
40
30
20
10
0
1
25
50
75
100
125
150
175
200
225
250
275
300
325
350
375
400
425
450
475
Größe vor Pruning
RandomJ48

Datensatz „segment“
http://repository.seasr.org/Datasets/UCI/arff/segment.arff
5000 Instanzen, 20 Attribute (19 numerisch + Klassenattribut)
Keine fehlenden Attributwerte

Unpruned J48:

Anzahl Blätter	**51**
Komplexität	**101**
Tiefe	**15**
Korrekt klassifizierte Instanzen	**97,0563%**

Pruned J48:

Anzahl Blätter	**39**
Komplexität	**77**
Tiefe	**13**
Korrekt klassifizierte Instanzen	**96,9264%**

100-mal jeweils 10-folds cross-validation Unpruned RandomJ48 bei steigender Komplexität

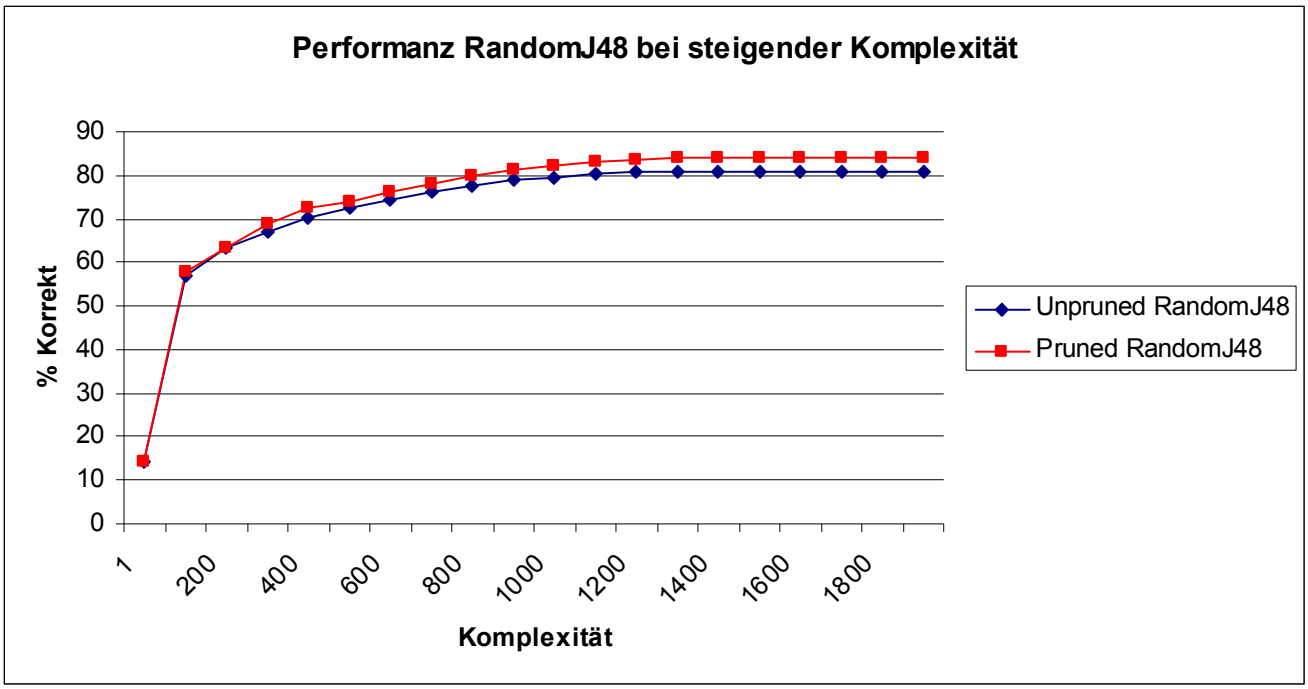
Kompl.	**1**	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**
% korrekt	14,29	56,94	63,16	67,13	70,11	72,60	74,25	76,10	77,52	78,87
Kompl.	**1000**	**1100**	**1200**	**1300**	**1400**	**1500**	**1600**	**1700**	**1800**	**1900**
% korrekt	79,41	80,29	80,69	80,79	80,94	80,78	80,75	80,93	80,85	80,81

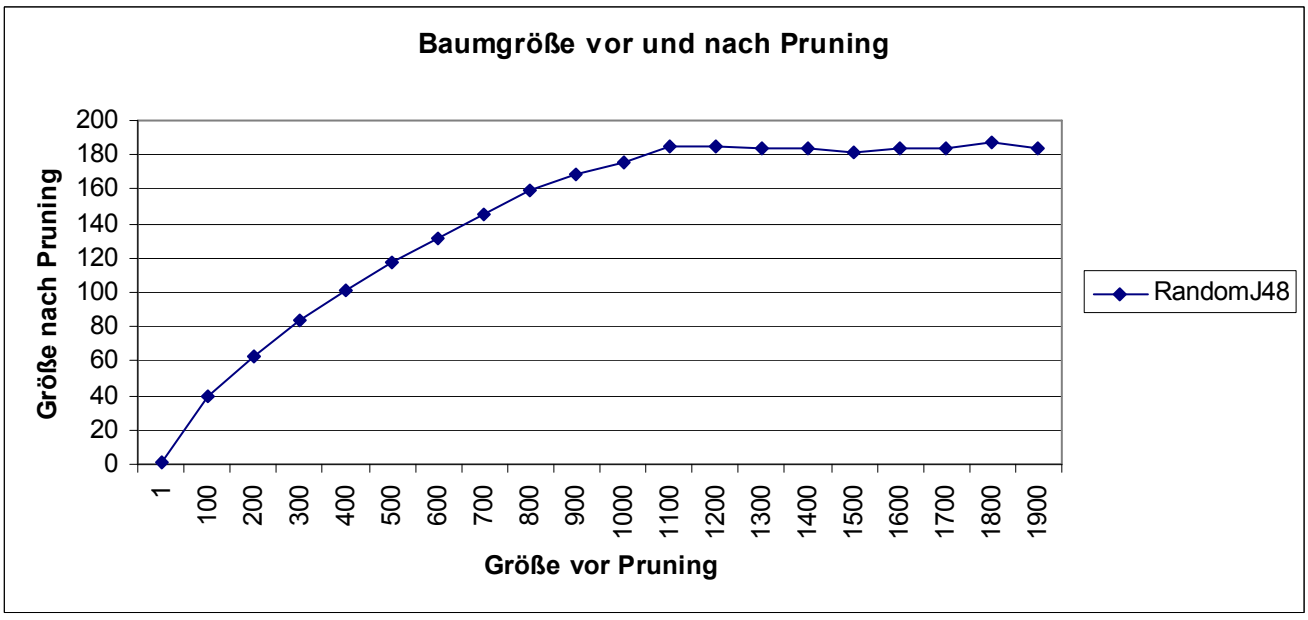
Kompl.	101	max
# Knoten		1120,99
% korrekt	57,15	80,72

100-mal jeweils 10-folds cross-validation Pruned RandomJ48 bei steigender Komplexität

Kompl.	**1**	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**
# Knoten	1	39,51	62,40	83,83	101,22	117,37	131,62	145,05	158,75	168,09
% korrekt	14,29	58,05	63,18	68,97	72,34	74,10	76,45	78,26	79,87	81,46
Kompl.	**1000**	**1100**	**1200**	**1300**	**1400**	**1500**	**1600**	**1700**	**1800**	**1900**
# Knoten	175,63	184,79	184,48	183,53	183,62	181,68	183,51	183,17	186,84	183,19
% korrekt	82,42	83,25	83,74	84,06	83,87	83,99	83,96	84,00	83,97	84,10

Kompl.	101	max
# Knoten	40,32	186,55
% korrekt	58,03	84,03


Performanz RandomJ48 bei steigender Komplexität
% Korrekt
90
80
70
60
50
40
30
20
10
0
1
200
400
600
800
1000
1200
1400
1600
1800
Komplexität
Unpruned RandomJ48
Pruned RandomJ48


Baumgröße vor und nach Pruning
Größe nach Pruning
200
180
160
140
120
100
80
60
40
20
0
1
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1700
1800
1900
Größe vor Pruning
RandomJ48

Datensatz „sonar“
http://repository.seasr.org/Datasets/UCI/arff/sonar.arff
208 Instanzen, 61 Attribute (60 numerisch + Klassenattribut)
Keine fehlenden Attributwerte

Unpruned J48:

Anzahl Blätter	**18**
Komplexität	**35**
Tiefe	**8**
Korrekt klassifizierte Instanzen	**69,7115%**

Pruned J48:

Anzahl Blätter	**18**
Komplexität	**35**
Tiefe	**8**
Korrekt klassifizierte Instanzen	**71,1538 %**

100-mal jeweils 10-folds cross-validation Unpruned RandomJ48 bei steigender Komplexität

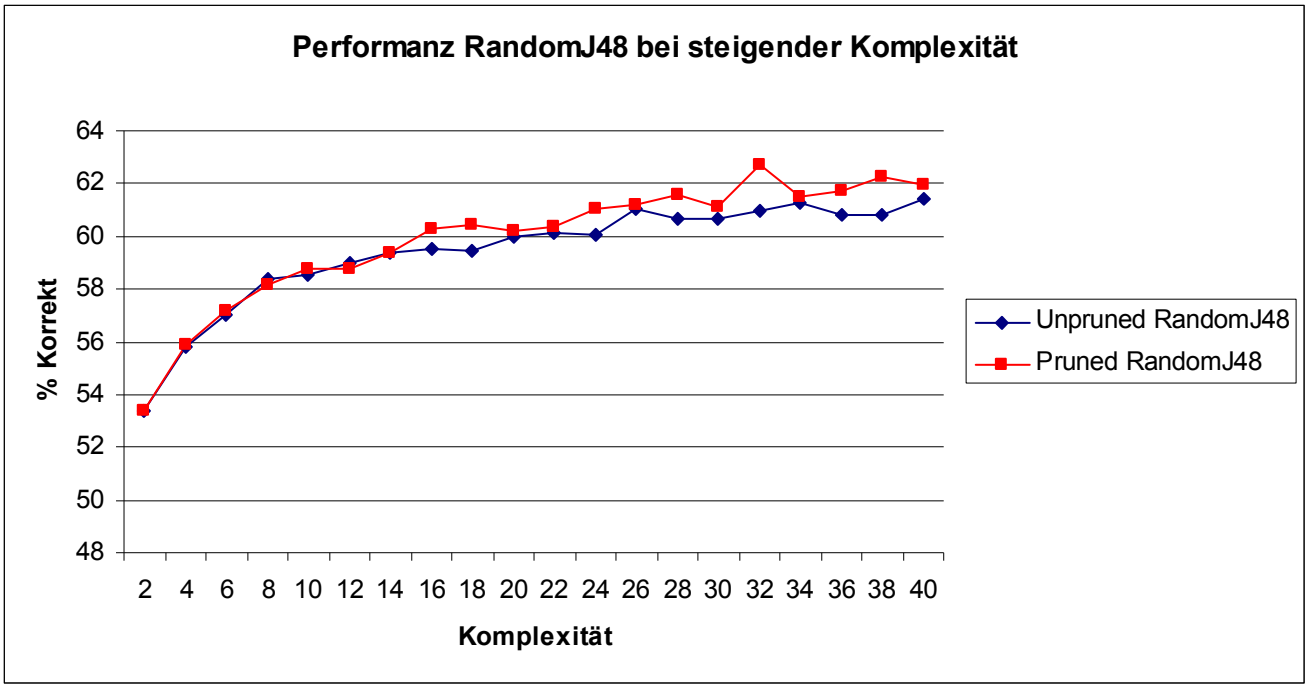
Kompl.	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**
% korrekt	53,38	55,80	57,01	58,40	58,52	58,98	59,37	59,52	59,47	60,01
Kompl.	**22**	**24**	**26**	**28**	**30**	**32**	**34**	**36**	**38**	**40**
% korrekt	60,11	60,02	61,03	60,68	60,64	60,97	61,28	60,85	60,82	61,41

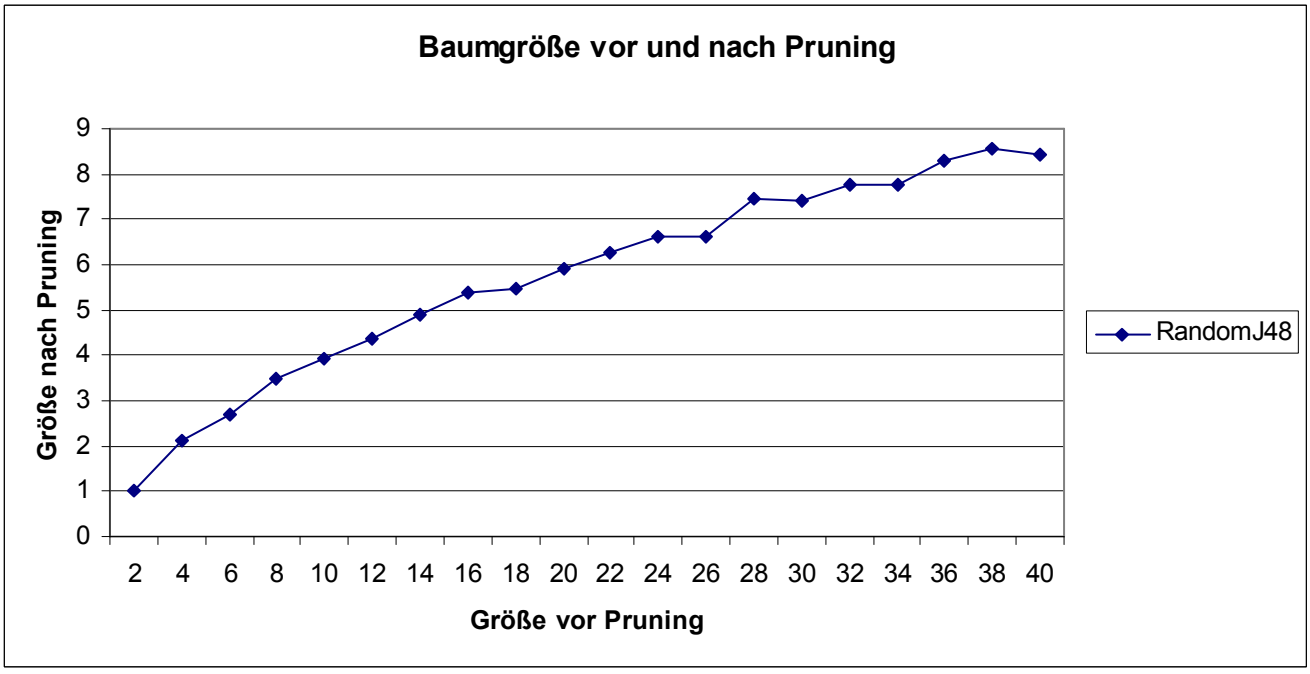
Kompl.	35	max
# Knoten		155,73
% korrekt	61,42	62,00

100-mal jeweils 10-folds cross-validation Pruned RandomJ48 bei steigender Komplexität

Kompl.	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**
# Knoten	1	2,10	2,68	3,47	3,93	4,37	4,89	5,40	5,48	5,90
% korrekt	53,38	55,91	57,20	58,16	58,73	58,78	59,38	60,28	60,41	60,22
Kompl.	**22**	**24**	**26**	**28**	**30**	**32**	**34**	**36**	**38**	**40**
# Knoten	6,26	6,62	6,63	7,47	7,42	7,78	7,76	8,31	8,57	8,42
% korrekt	60,38	61,07	61,19	61,58	61,09	62,69	61,53	61,72	62,27	61,92

Kompl.	35	max
# Knoten	8,18	21,75
% korrekt	62,05	65,00


Performanz RandomJ48 bei steigender Komplexität
% Korrekt
64
62
60
58
56
54
52
50
48
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
Komplexität
Unpruned RandomJ48
Pruned RandomJ48


Baumgröße vor und nach Pruning
Größe nach Pruning
9
8
7
6
5
4
3
2
1
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
Größe vor Pruning
RandomJ48

Datensatz „synthetic“
http://repository.seasr.org/Datasets/UCI/arff/kdd_synthetic_control.arff
600 Instanzen, 61 Attribute (60 numerisch + Klassenattribut), Index entfernt
Keine fehlenden Attributwerte

Unpruned J48:

Anzahl Blätter	**18**
Komplexität	**35**
Tiefe	**10**
Korrekt klassifizierte Instanzen	**91,6667%**

Pruned J48:

Anzahl Blätter	**18**
Komplexität	**35**
Tiefe	**10**
Korrekt klassifizierte Instanzen	**91,6667%**

100-mal jeweils 10-folds cross-validation Unpruned RandomJ48 bei steigender Komplexität

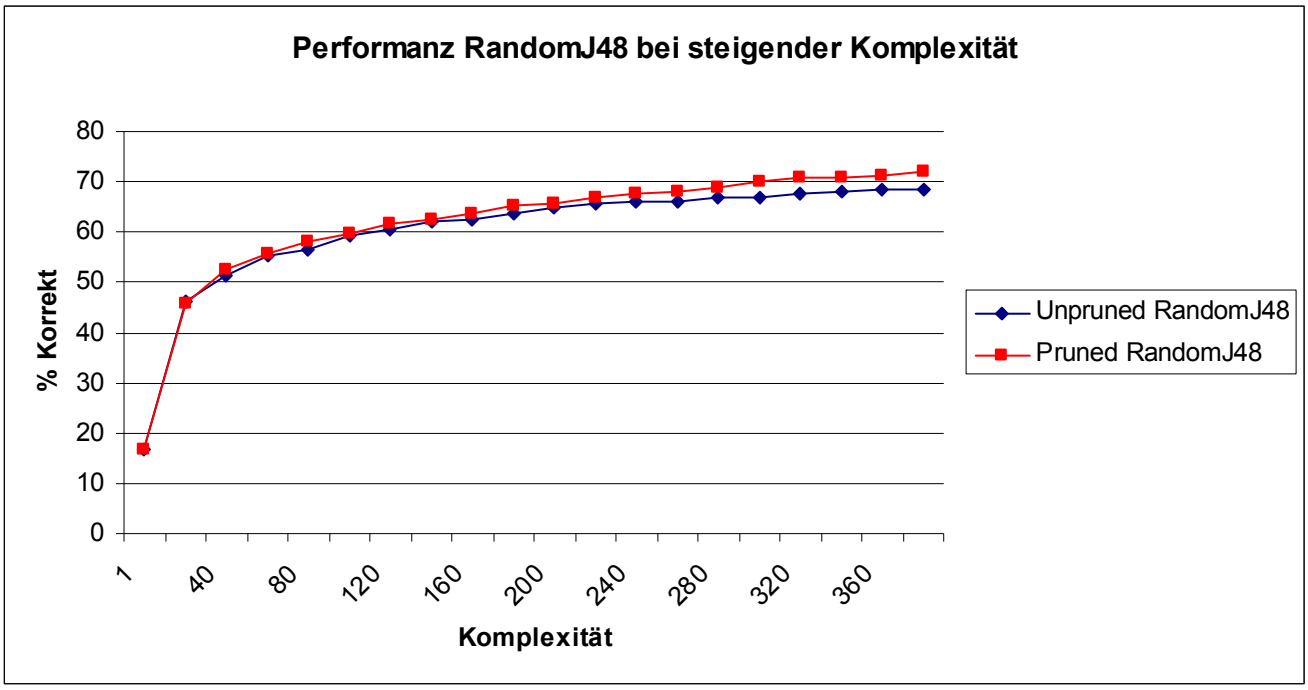
Kompl.	**1**	**20**	**40**	**60**	**80**	**100**	**120**	**140**	**160**	**180**
% korrekt	16,67	46,02	51,36	55,40	56,36	59,30	60,52	61,94	62,68	63,55
Kompl.	**200**	**220**	**240**	**260**	**280**	**300**	**320**	**340**	**360**	**380**
% korrekt	64,72	65,51	65,89	66,13	66,94	67,06	67,50	67,91	68,59	68,48

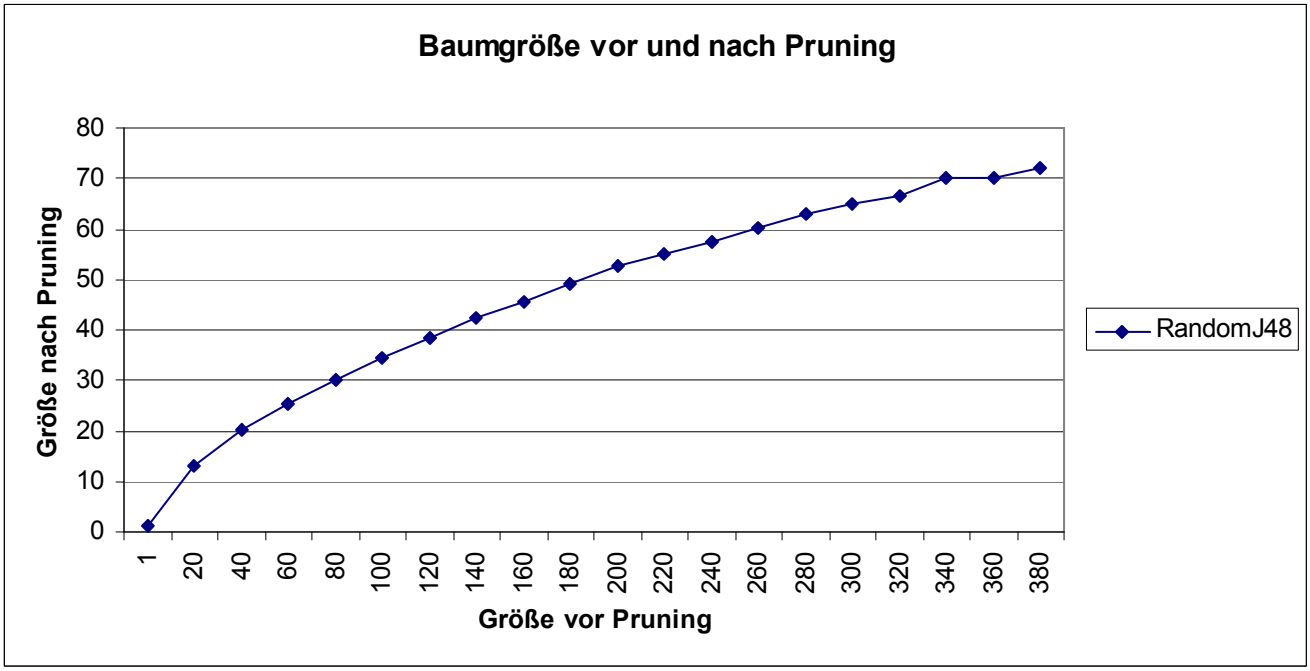
Kompl.	35	max
# Knoten		399,08
% korrekt	50,11	68,65

100-mal jeweils 10-folds cross-validation Pruned RandomJ48 bei steigender Komplexität

Kompl.	**1**	**20**	**40**	**60**	**80**	**100**	**120**	**140**	**160**	**180**
# Knoten	1	13,21	20,22	25,50	30,07	34,55	38,34	42,47	45,51	48,99
% korrekt	16,67	45,85	52,56	55,72	58,08	59,64	61,55	62,35	63,85	65,35
Kompl.	**200**	**220**	**240**	**260**	**280**	**300**	**320**	**340**	**360**	**380**
# Knoten	52,77	55,12	57,38	60,36	62,89	64,79	66,52	69,96	69,96	72,03
% korrekt	65,49	66,68	67,52	68,14	68,90	70,09	70,96	71,00	71,29	72,10

Kompl.	35	max
# Knoten	18,23	72,99
% korrekt	50,98	72,65


Performanz RandomJ48 bei steigender Komplexität
% Korrekt
80
70
60
50
40
30
20
10
0
1
40
80
120
160
200
240
280
320
360
Komplexität
Unpruned RandomJ48
Pruned RandomJ48


Baumgröße vor und nach Pruning
Größe nach Pruning
80
70
60
50
40
30
20
10
0
1
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
360
380
Größe vor Pruning
RandomJ48

Datensatz „tictactoe“
http://repository.seasr.org/Datasets/UCI/arff/tic-tac-toe.arff
958 Instanzen, 39 Attribute (9 nominell + Klassenattribut)
Keine fehlenden Attributwerte

Unpruned J48:

Anzahl Blätter	**139**
Komplexität	**208**
Tiefe	**7**
Korrekt klassifizierte Instanzen	**85,3862 %**

Pruned J48:

Anzahl Blätter	**95**
Komplexität	**142**
Tiefe	**7**
Korrekt klassifizierte Instanzen	**84,5511%**

100-mal jeweils 10-folds cross-validation Unpruned RandomJ48 bei steigender Komplexität

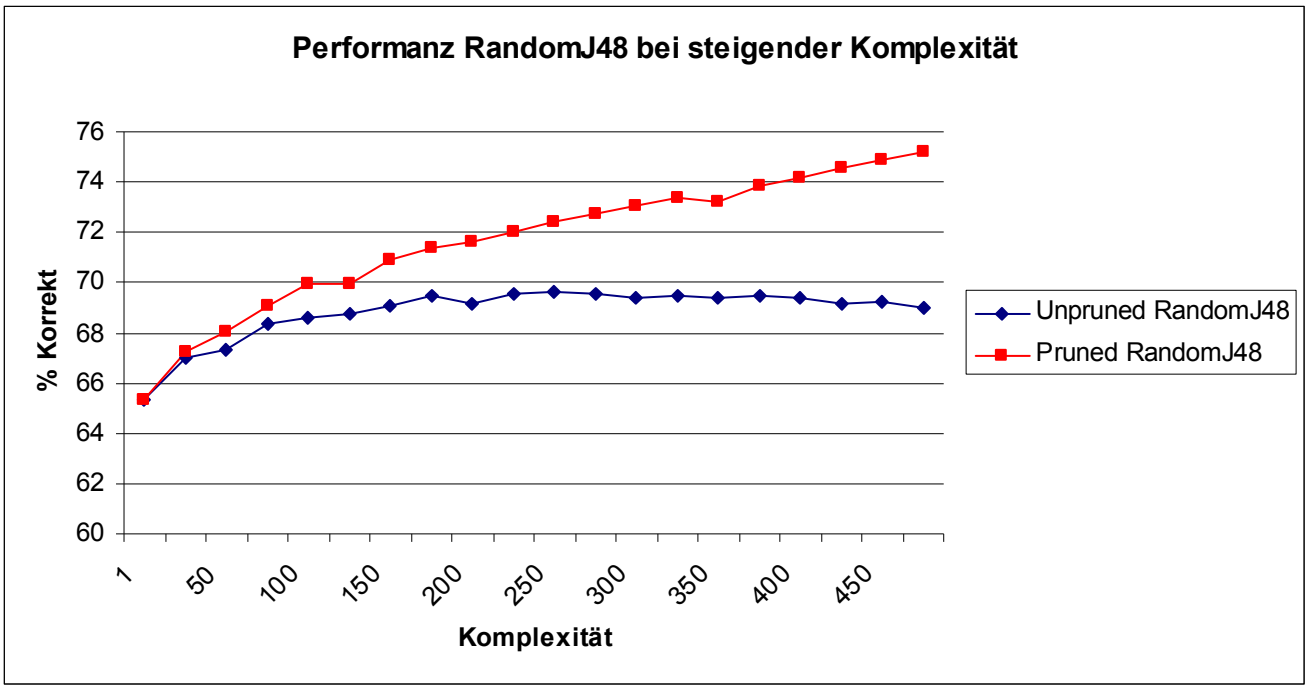
Komp.	**1**	**25**	**50**	**75**	**100**	**125**	**150**	**175**	**200**	**225**
% korrekt	65,34	67,03	67,32	68,36	68,56	68,72	69,07	69,48	69,19	69,56
Komp.	**250**	**275**	**300**	**325**	**350**	**375**	**400**	**425**	**450**	**475**
% korrekt	69,65	69,59	69,40	69,50	69,37	69,49	69,41	69,17	69,24	68,98

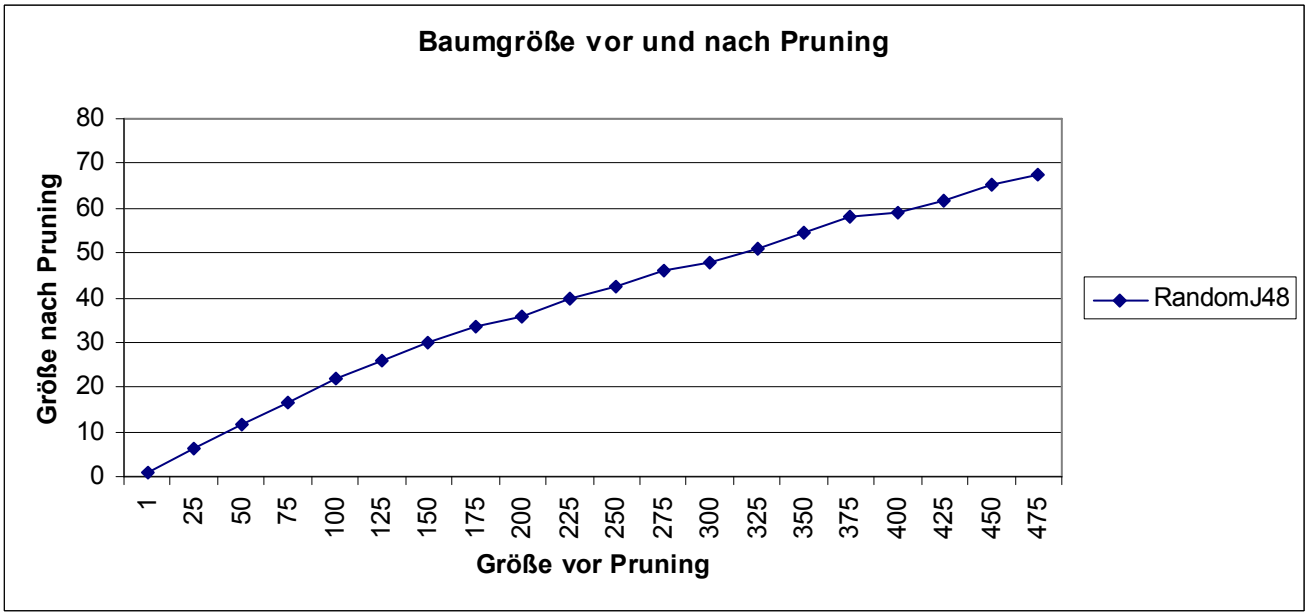
Komp.	208	max
# Knoten		551,97
% korrekt	69,36	69,19

100-mal jeweils 10-folds cross-validation Pruned RandomJ48 bei steigender Komplexität

Komp.	**1**	**25**	**50**	**75**	**100**	**125**	**150**	**175**	**200**	**225**
# Knoten	1	6,34	11,58	16,69	21,92	25,96	29,85	33,46	35,88	39,71
% korrekt	65,34	67,24	68,03	69,05	69,93	69,97	70,93	71,39	71,59	72,04
Komp.	**250**	**275**	**300**	**325**	**350**	**375**	**400**	**425**	**450**	**475**
# Knoten	42,46	46,02	48,00	50,97	54,35	58,13	59,01	61,63	65,34	67,64
% korrekt	72,43	72,71	73,02	73,37	73,23	73,82	74,13	74,59	74,89	75,20

Komp.	208	max
# Knoten	37,90	73,42
% korrekt	71,91	76,16


Performanz RandomJ48 bei steigender Komplexität
% Korrekt
76
74
72
70
68
66
64
62
60
1
50
100
150
200
250
300
350
400
450
Komplexität
Unpruned RandomJ48
Pruned RandomJ48


Baumgröße vor und nach Pruning
Größe nach Pruning
80
70
60
50
40
30
20
10
0
1
25
50
75
100
125
150
175
200
225
250
275
300
325
350
375
400
425
450
475
Größe vor Pruning
RandomJ48

Datensatz „vote“
http://repository.seasr.org/Datasets/UCI/arff/vote.arff
435 Instanzen, 22 Attribute (7 numerisch + 14 nominell + Klassenattribut)
Einige fehlende Attributwerte

Unpruned J48:

Anzahl Blätter	**19**
Komplexität	**37**
Tiefe	**9**
Korrekt klassifizierte Instanzen	**96,3218%**

Pruned J48:

Anzahl Blätter	**6**
Komplexität	**11**
Tiefe	**6**
Korrekt klassifizierte Instanzen	**96,3218%**

100-mal jeweils 10-folds cross-validation Unpruned RandomJ48 bei steigender Komplexität

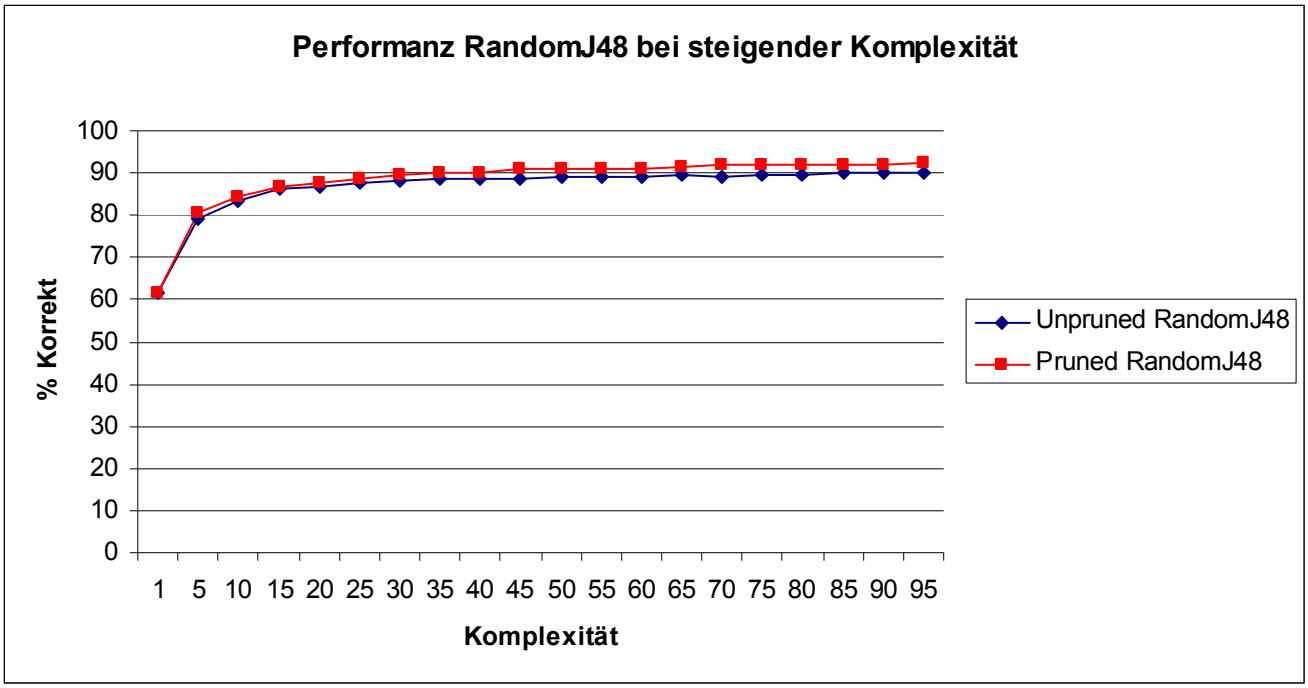
Komp.	**1**	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**
% korrekt	61,38	79,01	83,21	86,27	86,70	87,54	88,09	88,44	88,57	88,62
Komp.	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**	**90**	**95**
% korrekt	88,90	89,14	89,20	89,59	89,27	89,57	89,47	90,02	89,91	89,92

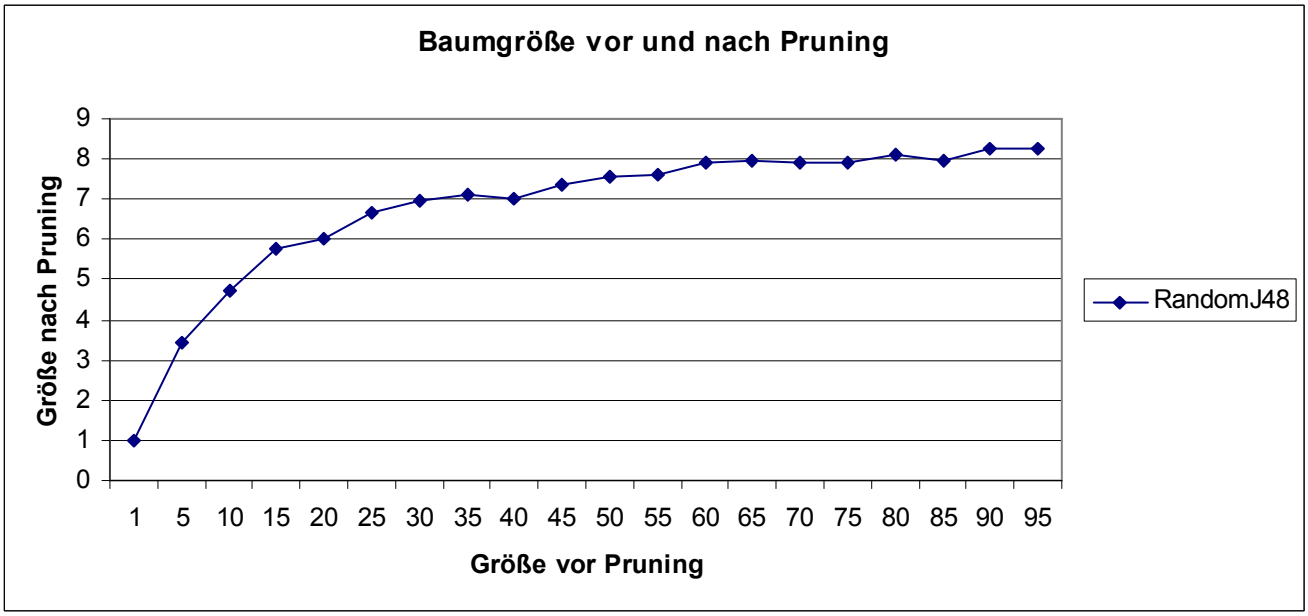
Komp.	37	max
# Knoten		240,72
% korrekt	88,42	90,57

100-mal jeweils 10-folds cross-validation Pruned RandomJ48 bei steigender Komplexität

Komp.	**1**	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**
# Knoten	1	3,41	4,73	5,75	6,04	6,68	6,95	7,10	7,01	7,38
% korrekt	61,38	80,37	84,34	86,52	87,75	88,77	89,53	89,91	90,03	90,79
Komp.	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**	**90**	**95**
# Knoten	7,57	7,59	7,93	7,96	7,93	7,91	8,11	7,95	8,26	8,25
% korrekt	90,97	91,07	91,21	91,27	91,71	91,95	91,85	92,16	92,11	92,23

Komp.	37	max
# Knoten	7,03	7,62
% korrekt	89,98	94,37


Performanz RandomJ48 bei steigender Komplexität
% Korrekt
100
90
80
70
60
50
40
30
20
10
0
1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95
Komplexität
Unpruned RandomJ48
Pruned RandomJ48


Baumgröße vor und nach Pruning
Größe nach Pruning
9
8
7
6
5
4
3
2
1
0
1 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95
Größe vor Pruning
RandomJ48