

Universität Kassel
Fachbereich Informatik/FB16

Projektarbeit

im Studiengang Informatik/Bachelor

Projekt: Data-Mining-Cup 2007

eingereicht von: Christian Voigtmann <voigtmannc@gmx.de>

eingereicht von: Martin Göb <martingoeb@gmx.de>

eingereicht von: Stefan Achler <stefan.achler@gmx.de>

eingereicht am: 21. Juni 2007

Betreuer: Dr. Andreas Hotho

Inhaltsverzeichnis

1	Einleitung	1
2	Aufgabenstellung	1
3	Datenanalyse & Preprocessing	2
4	Baseline	5
5	Lösungsansätze	5
5.1	SVM Zwei-Klassenproblem	5
5.2	Dreiklassenproblematik	6
5.2.1	Neuronale Netze	6
5.2.2	Entscheidungsbäume	6
5.2.3	Costensitive Classifierer	7
5.2.4	Verwendung von eindeutigen Regeln	8
5.2.5	Generalisierung der Regeln	9
5.3	Vierklassenproblem	10
5.3.1	Anwendung des J-48 Baum auf das Vierklassenproblem	11
6	Eingereichte Lösungen	11
7	Tools	11
8	Résumé	12

1 Einleitung

Im Rahmen eines Projektes an der Universität Kassel im Fachgebiet Wissensverarbeitung ¹, haben wir an dem Data-Mining-Cup 2007 ² teilgenommen. Der Data-Mining-Cup (DMC) ist ein Wettbewerb, welcher jedes Jahr von der TU Chemnitz ³ und der Firma prudsys ⁴ ausgerichtet wird. Dieser bietet Studenten und jungen Nachwuchsforschern die Möglichkeit Erfahrungen im Bereich des Data-Minings an Hand einer Aufgabenstellung aus der Praxis zu sammeln.

Wir gehen im Rahmen unserer Ausarbeitung als erstes auf die diesjährige Aufgabenstellung des Data-Mining-Cups ein. Anschließend nennen wir im Kapitel „Datenanalyse & Preprocessing“ die besonderen Eigenschaften der Datensätze und erläutern diese an Hand von Grafiken. Im Kapitel Baseline schaffen wir den Bewertungsmaßstab an dem wir unsere erzielten Lösungen messen. Kapitel 4 und 5 gehen auf die verwendeten Algorithmen und die daraus resultierenden Lösungen ein. Im Abschnitt Tools werden die verwendeten Softwarewerkzeuge genannt. Im Kapitel Résumé bewerten wir unsere erzielten Lösungsansätze.

2 Aufgabenstellung

Die diesjährige Aufgabe bestand aus einem überwachten Lernproblem. Es waren zwei Datensätze vorgegeben. Der erste Datensatz repräsentierte den Trainingsdatensatz mit Hilfe dessen ein Modell erlernt werden sollte, um den zweiten Datensatz den Classdatensatz zu klassifizieren.

Beide Datensätze sind auf den ersten Blick gleich aufgebaut. Beide beinhalten 50.000 Datensätze, wobei jeder Datensatz aus einer eindeutigen Kunden-ID und einer Sequenz aus Gutscheinen c01 bis c20 besteht. Jedem Kunden ist eine Gutscheinsequenz bzw. ein Kundenmuster gleicher Länge zugeordnet, beide Begriffe Gutscheinsequenz und Kundenmuster werden im Folgenden synonym verwandt. Aus der Sequenz von Gutscheinen geht hervor, wie oft ein Kunde die jeweiligen Gutscheine {c1, ..., c20} in der Vergangenheit eingelöst hat.

Der Trainingsdatensatz unterscheidet sich um eine zusätzliche Information von dem Classdatensatz. Die zusätzliche Information in dem Trainingsdatensatz gibt an, ob ein Kunde in Abhängigkeit seiner eingelösten Gutscheine aus der Vergangenheit einen weiteren Coupon einlösen würde oder nicht. Löst dieser einen weiteren Gutschein ein, ist hierbei noch zwischen einem höherwertigen Coupon „B“ und einem niederwertigen Coupon „A“ zu unterscheiden.

Das Ziel der Aufgabe besteht darin, aus dem Einlöseverhalten des Kunden bezüglich seiner Gutscheine aus der Vergangenheit ein Modell zu gewinnen, welches bei Anwendung auf den Classdatensatz zuverlässig bestimmen kann, ob ein Kunde einen weiteren Gutschein A, B einlöst oder nicht.

¹<http://www.kde.cs.uni-kassel.de/>

²www.data-mining-cup.de/

³<http://www.tu-chemnitz.de/>

⁴<http://www.prudsys.de/>

Es ist zu beachten, dass ein richtig zugewiesener Coupon A mit drei Punkten und ein richtig zugewiesener Coupon B mit sechs Punkten belohnt wird. Wird bestimmt, dass eine Kunde keinen weiteren Coupon einlösen würde, so erhält man keinen Punkt. Ordnet man hingegen einem Kunden den Coupon A oder B falsch zu, so ergibt sich daraus ein Strafpunkt.

Mit Hilfe der Gewinnfunktion

$$G(x) = 3 * AA + 6 * BB - 1 * (NA + NB + BA + AB)$$

lässt sich zum Schluss ermitteln, wie erfolgreich das eigene Modell auf den zu klassifizierenden Datensätzen gearbeitet hat.

3 Datenanalyse & Preprocessing

Die Spalten c01 bis c20 symbolisieren jeweils einen früheren Gutschein, der von einem Kunden in der Vergangenheit eingelöst wurde. Ein Kunde hat einen Gutschein maximal vier mal und minimal null mal eingelöst. Die Null kommt am häufigsten vor, gefolgt von der eins. Die Zahlen zwei, drei und vier dagegen sind seltener vertreten.

Abbildung 1 zeigt einen Ausschnitt der Trainingsdaten. Eine Zeile steht dabei für ein Kundenmuster.

ID	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	COUPON
97025	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	N
97032	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	A
97093	0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	1	0	0	0	0	B

Abbildung 1: Ausschnitt aus den Trainingsdaten

Die Verteilung der Coupons A, B und N ist nicht gleich verteilt. Insgesamt gibt es 8668 Kunden die einen A-Coupon einlösen, 3307 Kunden die ein B-Coupon einlösen und 38024 Kunden die keinen Coupon (N) einlösen. In Prozenten ausgedrückt heißt das, dass 76 % der Kunden keinen Coupon (N), 17 % eine A-Coupon und 7 % einen B-Coupon einlösen.

Es ist wichtig zu wissen, wie relevant die einzelnen Gutscheine für die Entscheidung am Ende sind. Hat jeder Gutschein den gleichen Einfluss auf die Wahl des Coupons oder gibt es Gutscheine, die eine höhere Gewichtung haben als andere? Um das herauszufinden haben wir uns die einzelnen Gutscheine genauer angeschaut. Dabei haben wir festgestellt, dass der Gutschein c06 von keinem Kunden eingelöst wurde und somit für die Entscheidungsfindung keine Bedeutung hat. Verdeutlicht wird diese Tatsache durch die Abbildungen 2. Auf der Abszisse sind die einzelnen Gutscheine aufgetragen und auf der Ordinate die Anzahl der eingelösten Gutscheine. Die Farben rot, beige und türkis stehen für die einzelnen Coupons und der blaue Balken repräsentiert die Gesamteinlösungen der einzelnen Gutscheine.

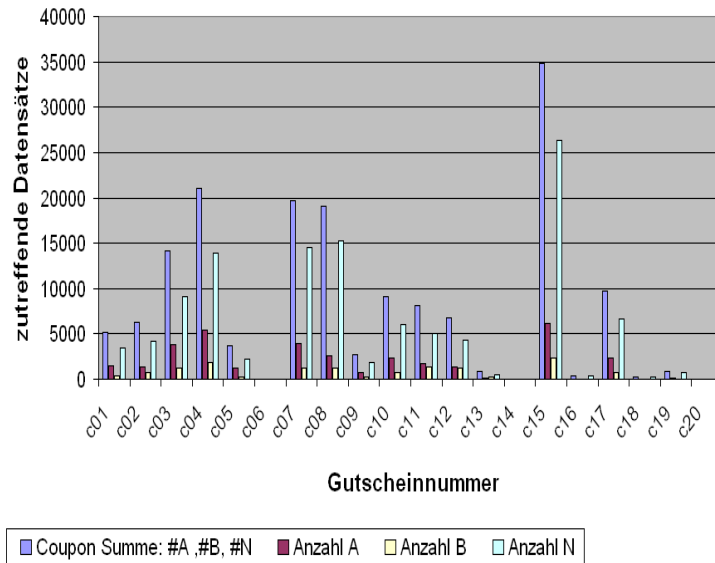


Abbildung 2: Gutscheineinlösungen

Es fällt zum Beispiel der Gutschein c15 auf, der von fast 35.000 Kunden mindestens einmal eingelöst wurde. Ebenso ist zu erkennen, dass die Zuordnungsverteilung von den einzelnen Gutscheinen unterschiedlich ist. Zum Beispiel ist der N-Anteil des Gutscheins c19 überdurchschnittlich hoch im Gegensatz zu Gutschein c05.

Wir haben nicht nur überflüssige Gutscheine gelöscht, sondern auch neue Attribute aus den vorhandenen Gutscheinen erstellt, um neue Informationen zu gewinnen.

Zu diesem Zweck haben wir zwei Algorithmen entworfen, die wir in Java implementiert haben. Der erste Algorithmus hat jeden Gutschein mit jedem Anderen addiert bzw. multipliziert. So entstehen jeweils 136 (c6, c14 und c20 wurden nicht berücksichtigt) neue Attribute.

Wir haben festgestellt, dass diese neuen Merkmale keinen Informationsgewinn bringen. Die Klassifikatoren haben keine besseren Ergebnisse geliefert. Des Weiteren haben verschiedene Attributfilter die meisten Attribute wieder gelöscht.

Der zweite Algorithmus berechnet die Anzahl der eingelösten Gutscheine pro Kunde. Auf diese Weise haben wir festgestellt, dass Kunden, welche einen, zwei oder drei Gutscheine eingelöst haben mit, überdurchschnittlich hoher Wahrscheinlichkeit keinen gewinnbringenden Coupon A oder B einlösen würden (siehe Abbildung 3).

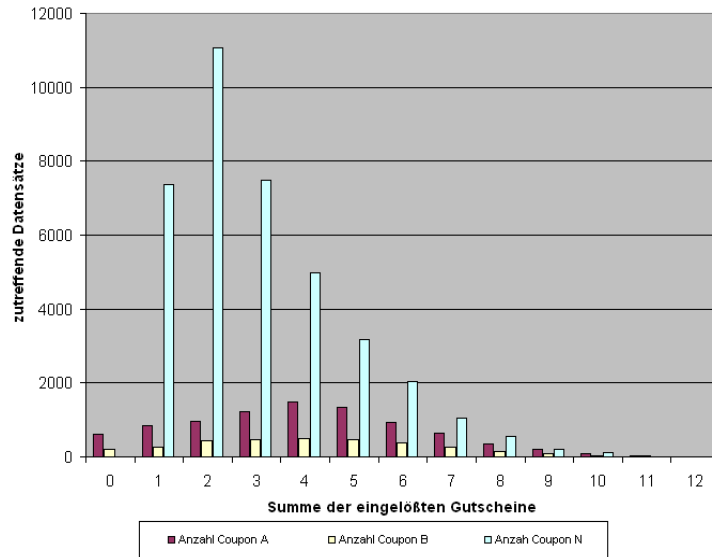


Abbildung 3: Zusammenhang zwischen eingelösten Gutscheinen und Coupons

Durch die Nichtberücksichtigung der Kunden, welche einen, zwei oder drei Gutscheine einlösen, hat sich die Verteilung der Klassen A, B und N zu unseren Gunsten verschoben, wie in Abbildung 4 dargestellt.

	Original (Prozent / Anzahl)	Reduziert (Prozent / Anzahl)
A	17 / 6840	28 / 4462
B	7 / 2649	11 / 1731
N	76 / 30511	61 / 9850

Abbildung 4: Verteilung der Coupons nach Reduzierung

Um unsere Ergebnisse zu überprüfen und um Overfitting zu vermeiden, haben wir den Trainingsdatensatz in 2 Datensätze aufgeteilt. Zum einen erstellten wir einen neuen Trainingsdatensatz mit 40.000 Einträgen (wenn im weiteren Dokument von einem Trainingsdatensatz gesprochen wird, ist der neue Datensatz gemeint) und zum anderen einen Holdoutdatensatz mit 10.000 Einträgen. Die 2 Datensätze sind disjunkt hinsichtlich ihrer IDs und die Auswahl der Datensätze war dabei zufällig. Auf dem Trainingsdatensatz haben wir die Classifier lernen lassen, um schließlich auf dem Holdoutdaten die erzeugten Modelle zu testen.

Interessant ist ebenfalls die Anzahl der verschiedenen Kundenmuster. Nimmt man vereinfacht an, dass ein Kunde einen Gutschein maximal einmal einlöst, ergeben sich daraus ca. eine Millionen verschiedene Muster. Wir haben auf den Trainingsdaten 2566 verschiedene Muster gefunden. 1611 Muster sind eindeutig einem Coupon zuzuweisen, 955 Muster sind mehrdeutig. Genaueres wird unter

dem Abschnitt „Verwendung von eindeutigen Regeln“ im Kapitel „Lösungsansätze“ erklärt.

4 Baseline

Um für unsere Lösungsansätze ein Bewertungskriterium zu schaffen, haben wir uns die folgenden Referenzwerte erzeugt: Auf den, nicht zum Lernen des Modells genutzten, 10.000 Houldoudaten wurde ermittelt, wie viele Kunden jeweils einen Coupon $\{A, B\}$ eingelöst haben. Daraus ergab sich die folgende Verteilung:

- A: 1828 Coupon Kunden, * 3 = Gewinn von 5484
- B: 658 Coupon Kunden, * 6 = Gewinn von 3948
- N: 7514 Coupon Kunden, * 0 = Gewinn von Null.

Klassifiziert man diese 10.000 Kunden richtig, so ist ein Gewinn von maximal 9432 möglich.

Um weitere Richtwerte zu erhalten, haben wir:

- Jedem Kunden einen A Coupon zugewiesen. Der Gewinn ist -4516.
- Jedem Kunden einen B Coupon zugewiesen. Der Gewinn ist -6502.
- Jedem Kunden keinen Coupon zugewiesen. Der Gewinn ist 0.
- Jedem Kunden zufällig einen Coupon zugeordnet, wodurch ein Gewinn -2683 von erzielt wird.
- Jedem Kunden einen Coupon gemäß der ungleichen Verteilung zugewiesen. Wir gehen davon aus, dass in dem Datensatz 76% N , 17% A und 7% B die Verteilung ist, so ist der Gewinn -10.

Mit diesem Wissen kann man nun etwas über die Güte der angewandten Modelle aussagen. Daraus ergibt sich für die gelernten Modelle die Minimalanforderung, dass diese auf dem Houldoudatensatz mindestens einen Gewinn von 0 erreichen.

5 Lösungsansätze

In diesem Kapitel werden kurz unsere Lösungsansätze vorgestellt, die wir im Laufe der Projektarbeit erarbeitet und evolutionär verbessert haben.

5.1 SVM Zwei-Klassenproblem

Eine Support-Vector-Maschine (SVM) ist ein Klassifikator, der in der Lage ist eine Menge von Objekten in zwei Klassen zu unterteilen. Die Trennung der Daten wird durch eine Hyperebene, die in den Raum gelegt wird vollzogen. Eine detaillierte Funktionsweise der SVM findet man hier [JST00]

Um eine SVM sinnvoll auf unsere 40.000 Trainingsdatensätze anwenden zu können, haben wir die Daten als erstes in zwei Klassen C und N unterteilt. Die Klasse C stellt hier die Vereinigung der Klassen A und B dar. Die Idee ist es, den Fokus zuerst auf die Erkennung der Klasse N zu legen, welches einfacher erschien, wenn man die Klassen A und B zusammen legt, da das Verhältnis von C zu N natürlich besser ist als das Verhältnis von A oder B zu N.

Tatsächlich war die SVM zum Beispiel mit dem Sigmuiden Kernel in der Lage, die Klasse N bezogen auf die 10.000 Holdout Daten 7151 mal richtig zu klassifizieren, sowie 63 C's richtig zu klassifizieren. Leider wurden aber auch 2128 C's als N's klassifiziert. Daraus lässt sich erkennen, dass die SVM so gut wie keine Trennung zwischen C und N vornahm, sondern knapp 93 % der Daten einfach der Klasse N zugewiesen hat. Das Ergebnis ist deswegen unbrauchbar, da gleiche Sequenzen aus den Trainingsdaten auf unterschiedliche Klassen C und N zeigen und somit die Platzierung einer entsprechenden Hyperebene zur Trennung der Daten im Raum nicht eindeutig möglich zu sein scheint.

5.2 Dreiklassenproblematik

Wie aus der Aufgabenstellung zu entnehmen ist, besteht die Aufgabe darin, Kunden in eine der drei möglichen Klassen einzuordnen. Algorithmen welche in der Lage sind eine Klassifizierung für drei Klassen vorzunehmen, werden im Folgenden beschrieben:

5.2.1 Neuronale Netze

Neuronale Netze versuchen die Informationsverarbeitung eines Gehirns nachzuahmen. Sie bieten sich für die Aufgabenstellung an, da sie robust gegenüber Abweichungen in den Eingabemustern sind. Wir haben die Implementierung eines Multilayer Perzeptron (MLP) und Backpropagation in Weka verwendet. Erklärungen für Neuronale Netze findet man unter anderem in [SJR03]. Die Kundenmuster weisen zu widersprüchliche Informationen auf. Aus diesem Grund kann das KNN keine sinnvolle Adaption der Gewichte in der Trainingsphase vornehmen.

5.2.2 Entscheidungsbäume

Ein probates Mittel, um Abhängigkeiten in Daten zu erkennen, sind Entscheidungsbäume. Entscheidungsbäume haben die Eigenschaft, dass sie einen Baum aufbauen und für jeden Knoten eine Entscheidung nach bestimmten Kriterien fällen. Weiterhin ist es möglich für eine beliebige Anzahl an Klassen Zuordnungen zu treffen. Im Gegensatz zu den SVMs, welche eine Trennung zwischen zwei Klassen vornehmen können. Deswegen bieten sich diese sowohl für die Drei-, als auch für die Vierklassenproblematik an. Wie diese Algorithmen theoretisch funktionieren kann unter anderem in [ME00] nachgeschlagen werden.

WEKA bietet verschiedene Typen von Entscheidungsbäumen. Mit den

geeigneten Parametern kann man die in Abbildung 5 zu sehende Klassifizierung erzeugen.

A	B	N	<-Ermittelter Coupon (Spalte) Eigentlicher Coupon (Zeile)
312	18	1498	A
81	28	549	B
221	46	7247	N

Abbildung 5: Ergebnis des J-48 Modells

Das Modell wurde auf den Trainingsdaten gelernt und auf den Houldoutdaten angewendet.

Wie man sehen kann, werden überproportional viele N Coupons vorhergesagt, obwohl diese Kunden eigentlich mit einem A oder B hätten klassifiziert werden müssen. Es muss also eine Möglichkeit gefunden werden, um die Klassifizierung zugunsten der A und B Klasse gewichten zu können. Mit diesem Ansatz beträgt der Gewinn 738.

5.2.3 Costensitive Classifierer

Möchte man die Klassifizierungen zu Gunsten einer oder mehrerer Klassen beeinflussen, bieten sich Costensitive Classifierer an. Als Parameter wird eine Kostenmatrix übergeben, welche die Gewichtungen für die Klassifizierungen enthält. Weitere Informationen sind unter anderem hier zu finden [Dom99]. Um mit den in Weka implementierten Algorithmen kostensensitiv klassifizieren zu können, stellt Weka Implementierungen von MetaCost und CostSensitive zur Verfügung. Die folgende Tabelle zeigt die Effekte der Costensitive Classifierer auf ausgewählten Algorithmen:

Algorithmus	Alleine	MetaCost	CostSensitiveClassifier
ZeroR	0	0	0
JRip	158	396	37
J48	573	1340	323
RandomForest	738	1254	492
RepTree	573	1486	365
DecisionStump	0	0	0
Bagging (RandomForest)	724	1219	726
Bagging (J48)	601	1433	601
MultiBoost (DecisionStump)	642	-5394	0
NaiveBayes	664	571	449
NaiveBayesUpdateable	508	571	459

Abbildung 6: Auswirkungen von MetaCost und CostSensitiveClassifier

Erzielter Gewinn unter Berücksichtigung von MetaCost- und CostSensitiveClassifierer auf dem Houldoutdatensatz.

5.2.4 Verwendung von eindeutigen Regeln

Ein weiterer Ansatz zur Klassifizierung der Daten war die Bestimmung von eindeutigen Regeln in den Trainingsdaten. Die Idee war es eindeutige Regeln als fix zu markieren und ihrer Klassenzugehörigkeit blind zu vertrauen. Die eindeutigen Gutscheinsequenzen sollten dann im Zuge des Preprocessings [Lar06] auf die zu klassifizierenden Daten angewendet werden, indem überprüft wird, ob diese eindeutigen Sequenzen auch in den zu klassifizierenden Daten vorkommen. War das der Fall, wurde der gefundenen Sequenz die Klasse der eindeutigen Sequenz aus dem Trainingsdatensatz zugewiesen. Dazu haben wir mittels Perl und Java die Trainingsdaten analysiert und die Gutscheinsequenzen extrahiert, die eindeutig sind. In den 40.000 Trainingsdaten gab es folgende Verteilung von atomaren Gutscheinsequenzen auf die Klassen A, B, N.

	Coupon A	Coupon B	Coupon N
Eindeutige Gutscheinsequenzen in Trainingsdaten	345	169	1048

Abbildung 7: Eindeutige Gutscheinsequenzen in Trainingsdaten

Die eindeutig in den Trainingsdaten gefundenen Sequenzen für die Klassen A, B und N haben wir in den zu klassifizierenden 50.000 Datensätzen für Coupon A 290 mal, für Coupon B 118 mal und für Coupon N 1797 mal gefunden. Dabei ist zu beachten, dass wir uns nicht sicher sein konnten, ob die 290 wieder gefundenen Sequenzen in den Class Daten wirklich die Klasse A beschreiben. Analog verhält sich das zu den eindeutigen Sequenzen der Klassen B und N. Um diesen Ansatz zu verifizieren, haben wir auf den 10.000 Holdout Daten getestet, wie oft die 345 gefundenen eindeutigen Sequenzen aus den Trainingsdaten für die Klasse A wieder zu finden sind, und ob diese dann ebenfalls auf die Klasse A verwiesen und nicht etwa auf B oder N. Das Selbe gilt für die eindeutigen Sequenzen die auf die Klassen B bzw. N verweisen.

Dabei haben wir festgestellt, dass von den 345 eindeutigen Sequenzen für die Klasse A nur noch 19 von ihnen auf den Holdoutdatensätzen eindeutig waren. Dass heißt 326 Sequenzen kamen entweder gar nicht erst vor oder zeigten auf die Klassen B oder N. Für die 169 eindeutigen Sequenzen für Klasse B waren auf den Holdout Daten nur noch 3 eindeutig und für die Klasse N waren schließlich nur noch 132 Sequenzen eindeutig.

Wir haben uns schließlich gegen die Verwendung der eindeutigen Sequenzen im Preprocessingschritt entschieden, da nach der Verifizierung der eindeutigen Sequenzen auf den Holdout Daten nur noch 5.8 % der atomaren Sequenzen die auf Klasse A zeigen eindeutig waren, für Klasse N 12.59 % und für Klasse B sogar nur noch 1.77 %.

5.2.5 Generalisierung der Regeln

Ein weiterer Lösungsansatz ist die Erstellung von generalisierten Regeln zur Klassifizierung der Gutscheinsequenzen. Dazu haben wir „SIGOA“ (Standard Interface for Global Optimization Algorithms) ⁵ benutzt.

Mittels der genetischen Algorithmen sind wir in der Lage Classifier Systems [Wei07] beruhend auf den 40.000 Trainingsdatensätzen zu erzeugen. Ein Classifier System besteht aus einer Liste von Regeln. Eine Regel setzt sich aus einer Bedingung für jedes Attribut c_{01} , ..., c_{17} und aus einer Klasse, welche die Gutscheinsequenz klassifiziert, zusammen.

Abbildung 8 stellt einen Auszug aus einem Classifier System dar und zeigt exemplarisch eine Regel, die zu einer Gutscheinsequenz passt.

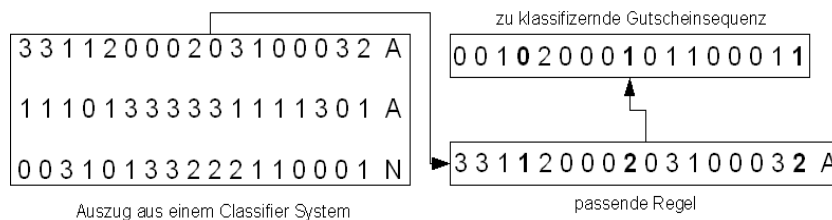


Abbildung 8: Auszug aus einem Classifier System

Eine Regel passt zu einem Datensatz, wenn keine ihrer Bedingungen den zu klassifizierenden Datensatz verletzt. Wird eine Regel entdeckt, die auf eine Sequenz genau passt, wird die Sequenz der entsprechenden Klasse zugeordnet. Gibt es für eine Sequenz keine passende Regel, so wird ihr die Klasse N zugewiesen. Um verrauschte Daten sicher klassifizieren zu können, passen auch Regeln auf eine Sequenz, die bis zu drei Fehler aufweisen.

Wie aus Abbildungen 8 ersichtlich, passt die generalisierte Regel trotz ihrer drei fehlerhaften Bedingungen (fett markiert) auf die zu klassifizierende Sequenz. Passen mehrere Regeln aus einem Classifier System auf eine Sequenz, so wird die Regel zur Klassifizierung verwendet, welche die geringsten Fehler aufweist und als erstes gefunden wurde.

Die einzelnen Bedingungen der Regeln sind Zahlen im Bereich 0 bis 3, wie bereits in Abbildung 8 ersichtlich. Dabei haben die einzelnen Werte die in Abbildung 9 dargestellten Bedeutungen. Ein Fehler bei der Anwendung einer Regel entspricht der Verletzung einer Bedingung durch ein korrespondierendes Attribut in der Gutscheinsequenz.

Abbildung 9 macht auch gleichzeitig die Kodierung der Classifier Systems (die ja letztendlich den Phenotyp des genetischen Algorithmus darstellen) in Genotypen deutlich. Die Struktur der Phenotypen ist optimal für eine Kodierung in binäre Strings geeignet, da sich sowohl die einzelnen Bedingung

⁵<http://www.sigoa.org/>

(mit je zwei Bit) als auch die drei Klassen (mit ebenfalls zwei Bit) direkt umrechnen lassen.

Der genetische Algorithmus wird von zwei Zielfunktionen gesteuert: die Gewinnfunktion führt zu profitablen Klassifikatoren. Zusätzlich wird die Anzahl der Regeln in den Classifier Systems minimiert, um Overfitting entgegen zuwirken.

Bedingung (Genotyp)	Bedingung (Phenotyp)	zugehöriger Attribut wert
00	0	muss 0 sein
01	1	muss ≥ 1
10	2	muss > 1
11	3	x element {0, 1, 2, 3}

Abbildung 9: Auszug aus einem Classifier System

Wir haben drei verschiedene Classifier für die Klassifizierung der Clasdaten erzeugt.

- Classifier I wurde auf den 40.000 Trainingsdatensätzen generiert.
- Classifier II wurde auf den 50.000 Trainingsdatensätzen generiert.
- Classifier III wurde auf den 50.000 Trainingsdatensätzen ohne Beachtung der Kunden generiert, welche insgesamt einen, zwei oder drei Gutscheine eingelöst haben.

Als Ergebnis erzielt Classifier I auf den 10.000 Holdoutdatensätzen einen Gewinn von 1719 Punkten, was in Bezug auf die Baseline das beste Ergebnis darstellt. Die Classifier II und III konnten wir nicht auf den Holdoutdatensätzen verifizieren, da diese aus den kompletten 50.000 Trainingsdaten generiert wurden.

5.3 Vierklassenproblem

Wie die Datenanalyse gezeigt hat, gibt es viele gleiche Kundenmuster, die sich nur im Coupon unterscheiden. Aus diesen Kundenmustern können die Algorithmen jedoch keine Regeln extrahieren. Betrachtet man jedes Kundenmuster (c01 bis c20) als Vektor, so sind diese Kunden alle über die gleichen Koordinaten beschrieben. Jedoch widersprechen sich die Muster in ihren Couponwerten.

Um die klassifizierenden Algorithmen zu unterstützen, wurden die nicht eindeutigen Kundenmuster mit einem neuen Coupon „W“ ausgezeichnet. Die Information, welche Muster nicht eindeutig sind, wurden anhand der 40 000 Daten getroffen, siehe Abschnitt Eindeutige Regeln. Auf diesen aufbereiteten Daten haben wir die Algorithmen erneut Modelle lernen lassen. Die als W klassifizierten Kunden werden in der Endabgabe mit N ausgezeichnet. Somit werden für diese Kunden keine Kosten verursacht.

5.3.1 Anwendung des J-48 Baum auf das Vierklassenproblem

Aufgrund des guten Ergebnisses des J-48 Baum, in Kombination mit dem MetaCost-Classifizierer auf den Dreiklassen, wurde dieser auf den Vierklassen verwendet. Die Validierung des gelernten J-48 Modells auf den Holdoutdatensatz zeigt eine Veränderung der Klassifizierungsverhältnisse. Diese Verbesserung ist nicht positiv. Dieses begründet sich damit, dass über 70 % aller Kunden in die Klasse W sortiert werden. Ungeachtet dessen verbesserte sich die Klassifizierung der gewinnbringenden A und B Kunden nicht.

6 Eingereichte Lösungen

Abgegeben haben wir den Classifier I und fünf weitere Abgaben sind durch die Kombination der Classifier I, II und III entstanden. Mittels der SQL-Datenbank haben wir alle IDs rausgefiltert, bei denen sich mindestens zwei Classifier uneinig sind. Dies sind 3634 Kunden.

Die Abgaben im einzelnen:

- Classifier I
- Allen 3634 Kunden ein A zugewiesen.
- Allen 3634 Kunden ein B zugewiesen.
- Allen 3634 Kunden ein N zugewiesen.
- Demokratie: Haben sich zwei Classifier für den selben Coupon entschieden, dann wurde dieser Coupon ausgewählt, ansonsten N.
- Optimismus: Entscheidet ein Classifier auf B, so wird als Coupon B ausgewählt, egal was die zwei anderen vorschlagen. Hat sich keiner für B entschieden wird als Coupon A gewählt.

7 Tools

Für die Bewältigung der diesjährigen Aufgabenstellung im Data-Mining-Cup habe wir auf mehrere Softwarewerkzeuge zurückgegriffen.

Für die Analyse der Daten und zur Visualisierung in Form von Diagrammen haben wir Microsoft Excel verwendet. SQL diente für die einfache und schnelle Suche nach Verhaltensmustern in den Daten. Für die Implementierung unsere Methoden und Algorithmen bedienen wir uns den Programmiersprachen Java und Perl.

Für die Bildung von Classifiern haben wir die Programme SVM Light ⁶ und Weka ⁷ benutzt, sowie (SIGOA) „Standard Inteface for Global Optimization Algorithms“ ⁸.

⁶<http://svmlight.joachims.org/>

⁷<http://www.cs.waikato.ac.nz/ml/weka/>

⁸<http://www.sigoa.org/>

8 Résumé

Unser Ziel ist eine Platzierung in dem oberen Drittel des DMC Rankings. Im Laufe des Projektes sind unerwartete Probleme aufgetreten. Wir stellten fest, dass wir mit den Standardalgorithmen unser Ziel nicht erreichen können. Die Punktzahl der in der Baseline vorgestellten Methoden konnten wir übertreffen, jedoch stellten uns die Ergebnisse nicht zufrieden. Mit Hilfe von kostensensitiven Classifieren konnten wir eine signifikante Steigerung der Punktzahl erreichen. Wir waren uns aber sicher, dass diese Verbesserung nicht ausreicht, um einen Platz im oberen Drittel zu belegen. Nach weiterer Recherche sind wir auf das Gebiet der „genetische Programmierung“ gestoßen. Da Weka keine eigene Implementierung dafür bereitstellt, fiel unsere Wahl auf „SIGOA“. Mit Hilfe von „SIGOA“ haben wir schließlich eine Möglichkeit gefunden, bessere Ergebnisse zu erzielen und so unserem Ziel näher zu kommen. Durch die Teilnahme am Data-Mining-Cup konnten wir einen interessanten und praxisnahen Einblick in den Bereich des Data-Minings bekommen. Ob wir unser Ziel letztendlich erreichen, erfahren wir durch die Veröffentlichung der Ergebnisse am 22. Juni 2007.

Literatur

- [Dom99] Pedro Domingos. *MetaCost: A general method for making classifiers cost-sensitive*. roceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, pp 155-164, 1999.
- [JST00] Nello Cristianini John Shawe-Taylor. *Support Vector Machines and kernel-based learning methods*. Cambridge University Press, 2000.
- [Lar06] Daniel T. Larose. *Data mining methods and models*. Hoboken, NJ : Wiley-Interscience, 2006.
- [ME00] Jörg Sander Martin Ester. *Knowledge Discovery in Databases*. Springer, 2000.
- [SJR03] Peter Norvig Stuart J. Russell. *Artificial intelligence : a modern approach*. Upper Saddle River, NJ : Prentice Hall, 2003, 2003.
- [Wei07] Thomas Weise. *Global Optimization Algorithms ? Theory and Application*. Thomas Weise, june 21, 2007 edition, Jun 2007. Online available at <http://www.it-weise.de/>.