RODM – Statistische Datenanalyse mit „R“ und Oracle Data Mining

Prof. Dr. Reinhold von Schwerin, Florian Langenbruch und Jörg Bellan
Hochschule Ulm

Schlüsselworte:

Data Mining, R, Datenanalyse, Statistik, ODM, CRISP-DM, Data Mining Cup

Einleitung

Die Gewinnung von wichtigen Informationen, etwa über das Kaufverhalten der Kunden, ist für Unternehmen ein wichtiger Erfolgsfaktor. Die Herausforderung hierbei ist das Auffinden von relevanten Informationen in den gesammelten Daten. Mit der Oracle Data Mining (ODM) Option, welche zusätzlich zur Enterprise Edition der Datenbank erworben werden kann, stehen dem Benutzer mächtige Algorithmen zur statistischen Datenanalyse zur Verfügung. Der Benutzer hat hierbei verschiedene Möglichkeiten diese Algorithmen zu verwenden. Neben der grafischen Oberfläche des *Oracle Data Miner Classic* können die Algorithmen auch direkt über eine PL/SQL API oder über eine Java API aufgerufen werden. Diese Möglichkeit wird von R-ODM, einem Paket für das unter der GNU General Public License stehende Statistiktool „R“, ausgenutzt und entsprechende Methoden bereitgestellt, welche die Data Mining Funktionalitäten der Oracle Datenbank verwenden können. Der Hauptvorteil von R-ODM besteht darin, dass auch Statistiker ohne besondere Datenbank- und SQL-Kenntnisse die Data Mining Methoden von Oracle in einer gewohnten Umgebung verwenden können.
Im folgenden Artikel werden die Erfahrungen im Umgang mit R-ODM anhand der Aufgabe des Data Mining Cups[1] 2010 der *prudsys AG* beschrieben. Anhand von Bestelldaten gilt es vorherzusagen, welche Kunden mit hoher Wahrscheinlichkeit wieder einkaufen und somit herauszufinden, welchen Kunden ein Gutschein geschickt werden sollte, um diese zu neuen Einkäufen zu animieren. Anhand des verbreiteten[2] *CRISP-DM*[3] Prozessmodells wird die schrittweise Umsetzung der statistischen Datenanalyse mit R-ODM beschrieben. Damit dient der Artikel auch als ein Leitfaden für den Einsatz von R-ODM.

Installation und Vorbereitung

Um die ODM Algorithmen aus der R Umgebung heraus aufrufen zu können, müssen zunächst die erforderlichen Pakete installiert werden. Neben dem eigentlichen R-ODM Paket muss noch das RODBC Paket installiert werden, welches Methoden bereitstellt, um mit Hilfe von ODBC Zugriff auf die Datenbank herzustellen. Dies kann bequem mit Hilfe einer Auswahlliste der RGui getan werden. Sämtliche Schritte werden automatisch ausgeführt. Voraussetzung für den Zugriff auf eine Oracle Datenbank ist, dass zum einen der Oracle Instant Client und zum anderen der entsprechende ODBC Treiber installiert ist. Hierbei muss zudem darauf geachtet werden, dass der Instant Client, ODBC Treiber und R einheitlich als 32-bit oder als 64-bit Variante installiert sein müssen. Der Installationsordner des Instant Client muss darüber hinaus die Datei *tnsnames.ora* umfassen, welche die eigentlichen Verbindungseinstellungen zur Datenbank enthält. Abschließend wird ein entsprechender ODBC Data Source Name (DSN) konfiguriert und hierzu der Eintrag der zuvor

[1] http://www.data-mining-cup.de
[2] http://www.kdnuggets.com/polls/2007/data_mining_methodology.htm
[3] http://www.crisp-dm.org

erstellten *tnsnames.ora* verwendet. Das RODM Paket kann nun in der R Console geladen (das Paket RODBC wird dabei automatisch geladen) und eine Verbindung zur Oracle Datenbank[4] hergestellt werden. Diese Verbindung wird dabei in einer Variablen gespeichert.

```
library(RODM)
DB = RODM_open_dbms_connection(dsn="rodm",uid="username",pwd="password")
```

Für R existieren zudem diverse grafische Oberflächen[5], welche die Arbeit erleichtern. Der Fokus kann dabei auch sehr unterschiedlich sein, zum Beispiel nimmt die Oberfläche *Rattle* dem Benutzer sämtliche Befehlseingaben ab, während die hier verwendete Software *Tinn-R* die Entwicklung und Verwaltung von R-Skripten unterstützt.

Datenanalyse in R

Im Anschluss an die erste Phase des CRISP-DM, dem Business Understanding, folgt das Datenverständnis, insbesondere das detaillierte Profiling der vorhandenen Daten. Bei dieser Aufgabe kann R seine Stärken als Statistik-Software ausspielen und bietet diverse Visualisierungsmöglichkeiten wie Histogramme oder Boxplots. Darüber hinaus lassen sich grundlegende statistische Werte wie Varianz oder Median sehr leicht berechnen. Obwohl sich diese Werte auch direkt mit SQL Statements oder mit Hilfe des Pakets DBMS_STAT_FUNCS der Oracle Datenbank ermitteln lassen, sind hierfür in R nur geringfügige Datenbankkenntnisse erforderlich. Um Daten betrachten zu können, müssen diese zunächst in der R Umgebung vorhanden sein. Hierfür werden Matrizen, im Zusammenhang mit R Data Frames genannt, verwendet. Mit R können zum einen Beispieldaten verwendet werden, Zufallszahlen generiert oder aber Daten aus der Datenbank abgefragt werden. Hierzu bietet das Paket RODBC entsprechende Methoden und mit folgendem Befehl wird der Inhalt einer Tabelle direkt in einem Data Frame gespeichert.

```
result <- sqlQuery(DB,"select * from myTable")
```

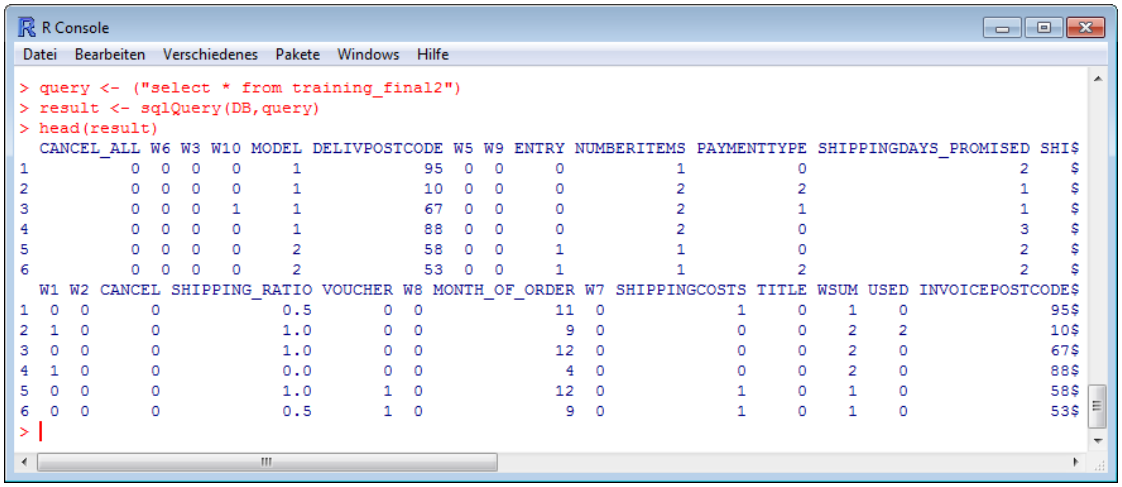

Abbildung 1: Anzeige des Inhalts einer Tabelle

[4] Anmerkung: Bei der Anwendung des Support Vector Machine Algorithmus stellte sich ein Fehler heraus, der in den Spracheinstellungen der Datenbank begründet ist. Die Fehlermeldung `ORA-40206: Ungültiger Einstellungswert für Einstellungsname SVMS_OUTLIER_RATE` taucht auf, wenn ein Komma als Dezimaltrennzeichen verwendet wird. Abhilfe schafft die Änderung des Parameters NLS_NUMERIC_CHARACTER, so dass ein Punkt als Trennzeichen akzeptiert wird.

[5] http://en.wikipedia.org/wiki/R_(programming_language)#Tools

Das Ergebnis, das im Datenframe `result` gespeichert ist, kann nun verarbeitet werden. Mit dem Befehl `head(result)` werden beispielsweise die ersten Zeilen inklusive der Spaltennamen dargestellt, wie in *Abbildung 1* zu sehen ist.

Die Beschreibung von Daten ist ein wesentlicher Teil der Statistik und umfasst eine systematische Suche nach aufschlussreichen Informationen über die Struktur der Daten. Um die zugrundeliegenden Daten besser verstehen zu können, werden mit statistischen Methoden die Daten möglichst genau beschrieben. Dazu gehört der Umfang des Datensatzes, die Identifizierung des Schlüsselattributs, die Anzahl der Attribute, der Datentyp der Attribute und die Anzahl der redundanten Datensätze. Für jedes Attribut müssen geeignete statistische Lagemaße und Streuungsmaße gebildet werden. R bietet hierzu einige Statistikfunktionen wie Maximum, Minimum, Mittelwerte und den Median an. Einen ersten Überblick über die Daten bietet die Funktion `summary()` aus der beschreibenden Statistik an. Die Funktion `summary()` listet für jede Tabellenspalte die Lageparameter Minimum, erstes Quartil (unteres Quartil), Median, Mittelwert, drittes Quartil (oberes Quartil) und Maximum auf. Diese Lageparameter können auch mit einem Box-Plot für einzelne Spalten visualisiert werden.

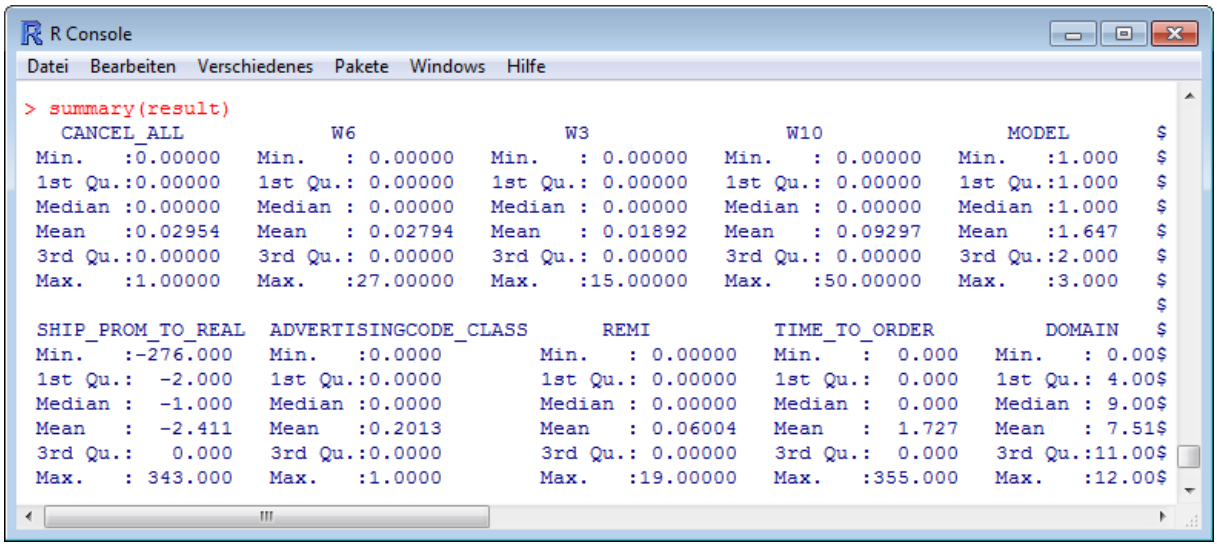

Abbildung 2: Statistische Werte mit Summary() Funktion

Zusätzlich zeigt die Funktion `str()` in R die Struktur eines Data Frames mit Anzahl des Datensatzes und die Anzahl der Variablen an. Diese Ansicht zeigt einen guten Überblick über die Daten und deren Beschaffenheit an (siehe *Abbildung 3*).

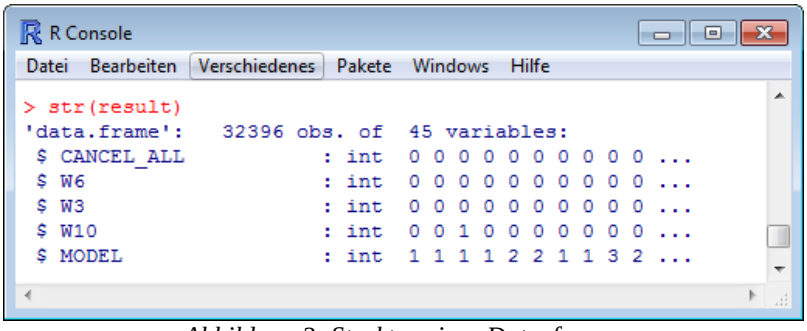

Abbildung 3: Struktur eines Datenframes

Zur Beschreibung von Daten werden in der deskriptiven Statistik grundsätzlich Maßzahlen eingesetzt, welche die Lage der Datenpunkte ausdrücken, eine Streuung erfassen, die Form der (Häufigkeits-) Verteilung beschreiben oder Zusammenhänge zwischen zwei Variablen untersuchen. Für alle Skalentypen ist es zu Beginn sinnvoll die Daten anhand der Häufigkeiten des Auftretens von Merkmalsausprägungen, also den einzelnen Datenpunkten einer Tabellenspalte, zu beschreiben. R bietet hierfür die Methode `table()` an, mit der eine Häufigkeitstabelle erstellt wird, zum Beispiel `absolut <- table(result$REMI)`. Wie zu sehen ist, erhält man mit dem Dollar-Zeichen direkten Zugriff auf die Spalte REMI des Data Frames. Zudem können mit folgenden Befehlen die relativen Häufigkeiten angezeigt werden.

```
relativ <- absolut / sum(absolut) #relative Häufigkeit
prozent <- relativ * 100; round(prozent, 1) #relative Häufigkeit gerundet
```

Um die Häufigkeiten auch visuell darzustellen, besteht in R die Möglichkeit mit relativ einfachen Aufrufen Torten- und Balkendiagramme zu erzeugen. Diese Diagramme können wie in nachstehendem Listing einzeln für eine Tabellenspalte erzeugt werden, oder auch aus mehreren Tabellenspalten bestehen, beispielsweise in einem Balkendiagramm. Zudem können mehrere Diagramme, auch von unterschiedlichen Diagrammtypen, in einem Output erzeugt werden.

```
pie(table(result$DOMAIN),main="Häufigkeiten von Domain", col=rainbow(20))
barplot(table(result$DOMAIN), main = "Häufigkeiten von Domain")
```

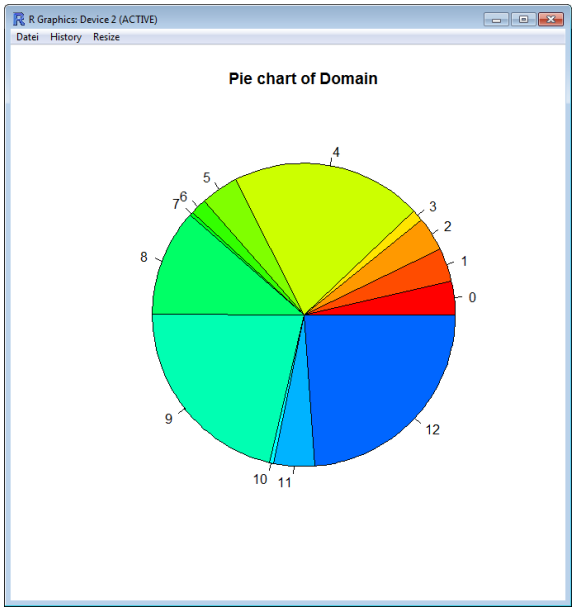

Abbildung 4: Pie Chart of Domain

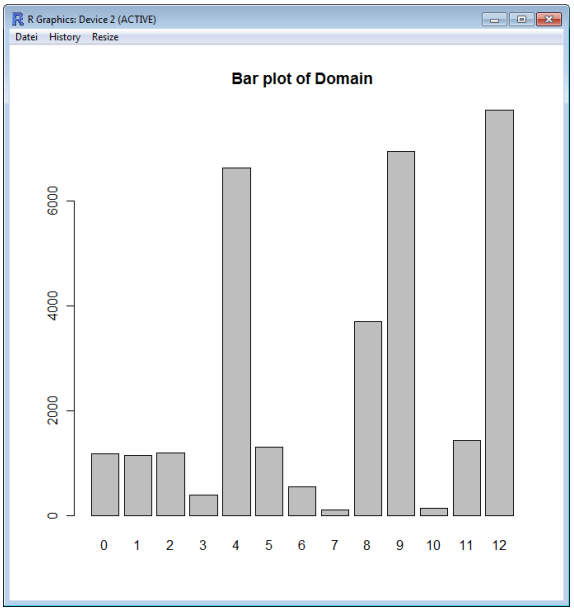

Abbildung 5: Bar plot of Domain

Grundsätzlich kann ein Data Frame mit dem Befehl `plot()` visualisiert werden. Die Komplexität eines Diagrammaufrufs steigt erst mit der Anzahl der optionalen Formatierungsparameter, die ein Diagrammtyp anbietet. Für nominal skalierte Daten sind die zuvor beschriebenen Methoden ausreichend, für ordinal skalierte oder metrische Daten erlaubt die Messbarkeit mit weiteren statistischen Methoden einen höheren Informationsgehalt. Ordinal skalierte Daten, wie auch metrische Daten, können sortiert und ein Rang gebildet werden. In R ist dies mit den simplen Befehlen `sort()` und `rank()` möglich. Weiter kann in R die Streuung der Daten betrachtete werden. Dazu kann neben der Spannweite (Range) auch die mittlere absolute Abweichung vom Median und die Median absolute Abweichung (Median-Deviation) gebildet werden.

```
MA <- mean(abs(result$DOMAIN - median(data$DOMAIN)));#Abweichung vom Median
D <- mad(result$DOMAIN, const=1); #Median-Deviation mit mad()
```

Eine grafische Darstellung von ordinal skalierten und metrischen Daten kann mit Punktdiagrammen (Dot-Plot) oder Box-Plots erfolgen. Mit der Funktion `stripchart()` kann in R ein Punktdiagramm gezeichnet werden, in dem die Beobachtungen als Punkte zu y-Werten in einem Koordinationssystem aufgetragen werden. Die x-Achse dient zur möglichen Unterteilung von mehreren Merkmalen, um diese damit vergleichen zu können.

```
stripchart(list(result$W6,result$W3,result$W2),vertical=TRUE,method="jitter
",jitter=0.1,group.names=c("W6","W3","W2"),ylim=c(0,15),main="Punktdiagramm
mit jitter")
```

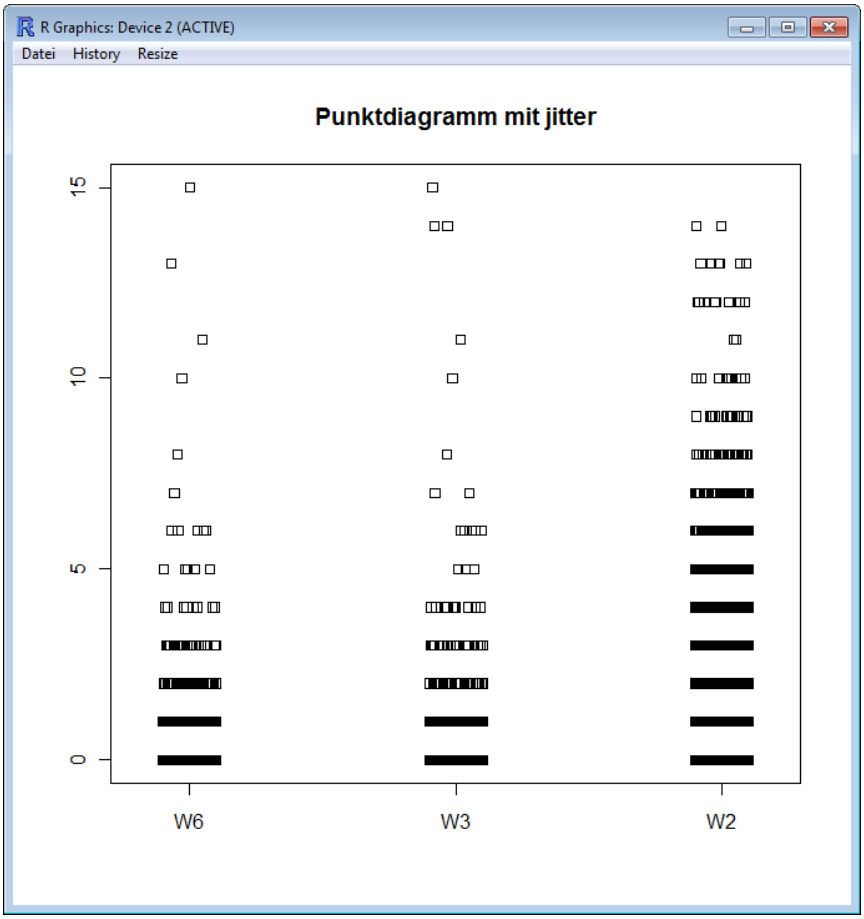

Abbildung 6: Punktdiagramm mit stripchart()

Mit der Methode *jitter* werden gleiche Werte im Punktdiagramm auf gleicher Höhe nebeneinander angeordnet, somit kann zudem die Häufigkeit eines Wertes dargestellt werden. Um mit R einen einfachen Boxplot eines Merkmals zu erzeugen, genügt der Befehl `boxplot(dataframe)`. Allerdings sind auch kompliziertere Grafiken mit einem höheren Informationsgehalt möglich. Beispielsweise kann die Streuung eines Merkmals mit Bezug auf das vorherzusagende Merkmal (hier TARGET90) visualisiert werden. Hierfür wird zuerst eine Unterreihe der ursprünglichen Daten mit einem Merkmal mit der Gruppierung nach dem Target („0“ oder „1“) gebildet.

```
ds <- rbind(data.frame(dat=data[,"WSUM"], grp="All"),
            data.frame(dat=data[,] [data$TARGET90=="0","WSUM"], grp="0"),
            data.frame(dat=data[,] [data$TARGET90=="1","WSUM"], grp="1"))
```

Anschließend wird der Boxplot mit dem erstellten Datenrahmen *ds* und der Gruppierungsformel *dat* erzeugt.

```
boxplot(formula=dat ~ grp, data=ds, col=rainbow(5), xlab="TARGET90", ylab="WSUM", ylim=c(0,10), notch=TRUE, main=" Boxplot - Distribution of WSUM nach TARGET90")
```

Zusätzlich wird noch der quadratische Mittelwert je Gruppe berechnet und als Punkt abgebildet.

```
points(1:3,summaryBy(dat ~ grp,data=ds,FUN=mean,na.rm=TRUE)$dat.mean,pch=8)
```

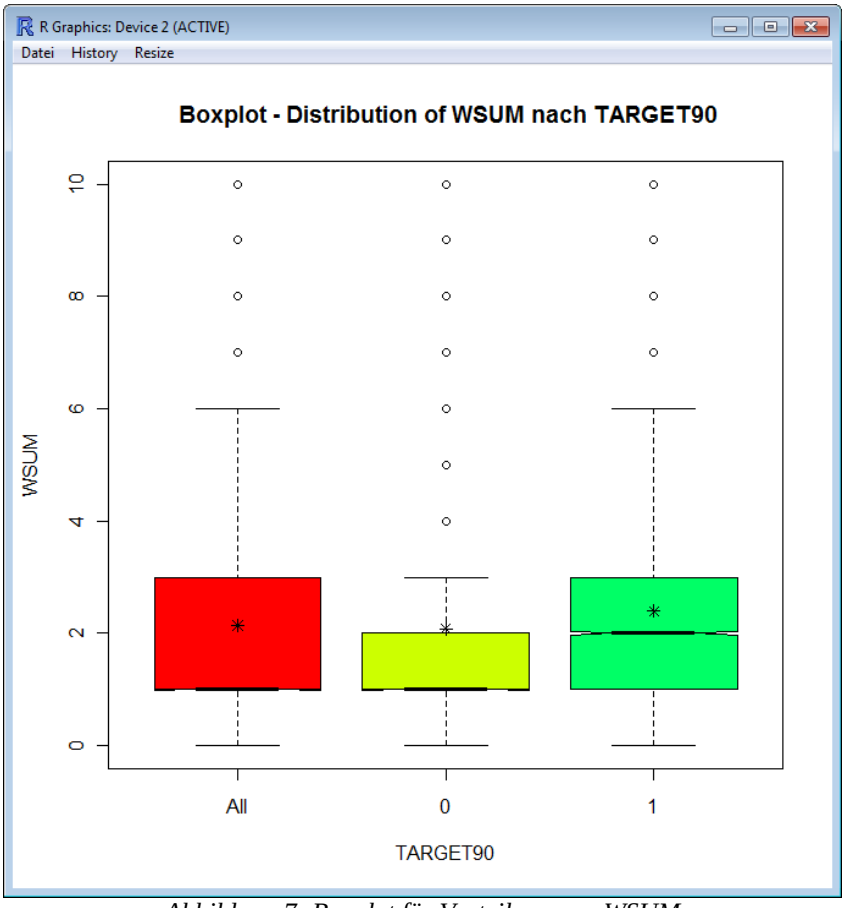

Abbildung 7: Boxplot für Verteilung von WSUM

Ergänzend zu den Maßzahlen und Verfahren für ordinal skalierte Daten bietet R weitere Möglichkeiten mehr Informationen aus den Merkmalen zu nutzen. Die aus der Statistik bekannten Methoden können in R sehr einfach auf die zu untersuchenden Daten angewendet werden. Dazu gehören der arithmetische Mittelwert, die Standardabweichung, die Varianz, das harmonische und geometrische Mittel. Auch kann durch Parametrisierung eines Balkendiagramms ein Fehlerbalkendiagramm erstellt werden um damit die graphische Darstellung von Mittelwerten und Standardabweichung zu ermöglichen.

Zu der umfangreichen Beschreibung und Erforschung der Daten besteht in der Data Understanding Phase nach CRISP-DM die Aufgabe darin, Zusammenhänge in den Daten zu finden. Eine geeignete Methode ist hierfür die Korrelationsanalyse. Da einige Data Mining Algorithmen die Unabhängigkeit von numerischen Eingabevariablen voraussetzen und Modelle mit unabhängigen Variablen bessere Ergebnisse liefern, ist eine vorherige Korrelationsanalyse sinnvoll. Beispielsweise kann mit der Funktion `plotcorr()` des Pakets *ellipse* eine Korrelationsanalyse vorgenommen werden. In folgendem Beispiel wird mit der Funktion `cor()` eine Korrelationsanalyse für die ersten fünf Variablen des Data Frames *result* durchgeführt.

```
cor <- cor(result[,1:5],use="pairwise",method="pearson")
# Die Korrelation nach ihrer Stärke einordnen.
ord <- order(cor[1,])
cor  <- cor[ord,ord]
plotcorr(cor,col=colorRampPalette(c("red","white","blue"))(11)[5*cor  +  6],
main="Korrelation anhand von Pearson")
```

Datenvorbereitung

Die Datenvorbereitung ist ein aufwändiger aber überaus wichtiger Schritt beim Data Mining. Hier kommt es jedoch auf den Anwender an, auf welche Art und Weise die Vorbereitung durchgeführt wird. Mit R kann die Anpassung des Data Frames mit Hilfe von Konstrukten wie IF-Abfragen und FOR-Schleifen geschehen und erfordert daher ebenso Fachkenntnisse wie die Lösung mit Hilfe von SQL Statements. Auch bei diesem Schritt kommt der Anwender allerdings ohne Datenbankkenntnisse aus.

Modellerstellung

Das hier verwendete Beispiel des Data Mining Cups stellt eine sogenannte Klassifikationsaufgabe dar. Oracle bietet hierzu vier Algorithmen: Naive Bayes, Decision Tree, Support Vector Machine und Logistic Regression. Um diese Algorithmen aufzurufen, bietet R-ODM entsprechende Methoden, welche schließlich das Package DBMS_DATA_MINING in der Datenbank verwenden. Hiervon bekommt der Anwender jedoch nichts mit. Die Data Mining Methoden werden in R-ODM ähnlich aufgerufen und verwendet wie bei der PL/SQL API. In der Dokumentation des RODM Pakets findet sich eine Auflistung und Beschreibung der möglichen Parameter, die beim Aufruf der Methode konfiguriert werden können. Das Entscheidungsbaummodell, um die hier gestellte Aufgabe zu lösen, lässt sich mit folgendem Befehl erstellen:

```
dt  <-  RODM_create_dt_model(database=DB,  data_table_name="TRAINING_DATA",
target_column_name="TARGET90", model_name="DTMODEL1")
```

Bei diesem Aufruf wurden nur die zwingend notwendigen Parameter verwendet und ansonsten die Standardwerte genutzt. An dieser Stelle sei auch darauf hingewiesen, dass der Modellname beachtet werden muss, da R-ODM im Zweifelsfall ohne Rückfrage vorhandene Modelle überschreibt. Ein wichtiger Parameter, der für sämtliche bereitgestellten Methoden verfügbar ist, ist `retrieve_outputs_to_R`. Dieser bestimmt, ob die Ergebnisse der Modellerstellung in den R Workspace zurückgeliefert werden oder nicht.
In *Abbildung 8* sind die Modelleinstellungen und die verwendeten Attribute dargestellt. Bei den Einstellungen lässt sich zudem schnell erkennen, ob für bestimmte Parameter vom Benutzer angegebene Werte oder die Standardwerte genutzt wurden. Wie bei sämtlichen Objekten in R, sind auch hier einzelne Bestandteile separat zugänglich. Zum Beispiel lässt sich mit dem Befehl `dt$model.model_settings` bewirken, dass nur die Modelleinstellungen angezeigt werden.

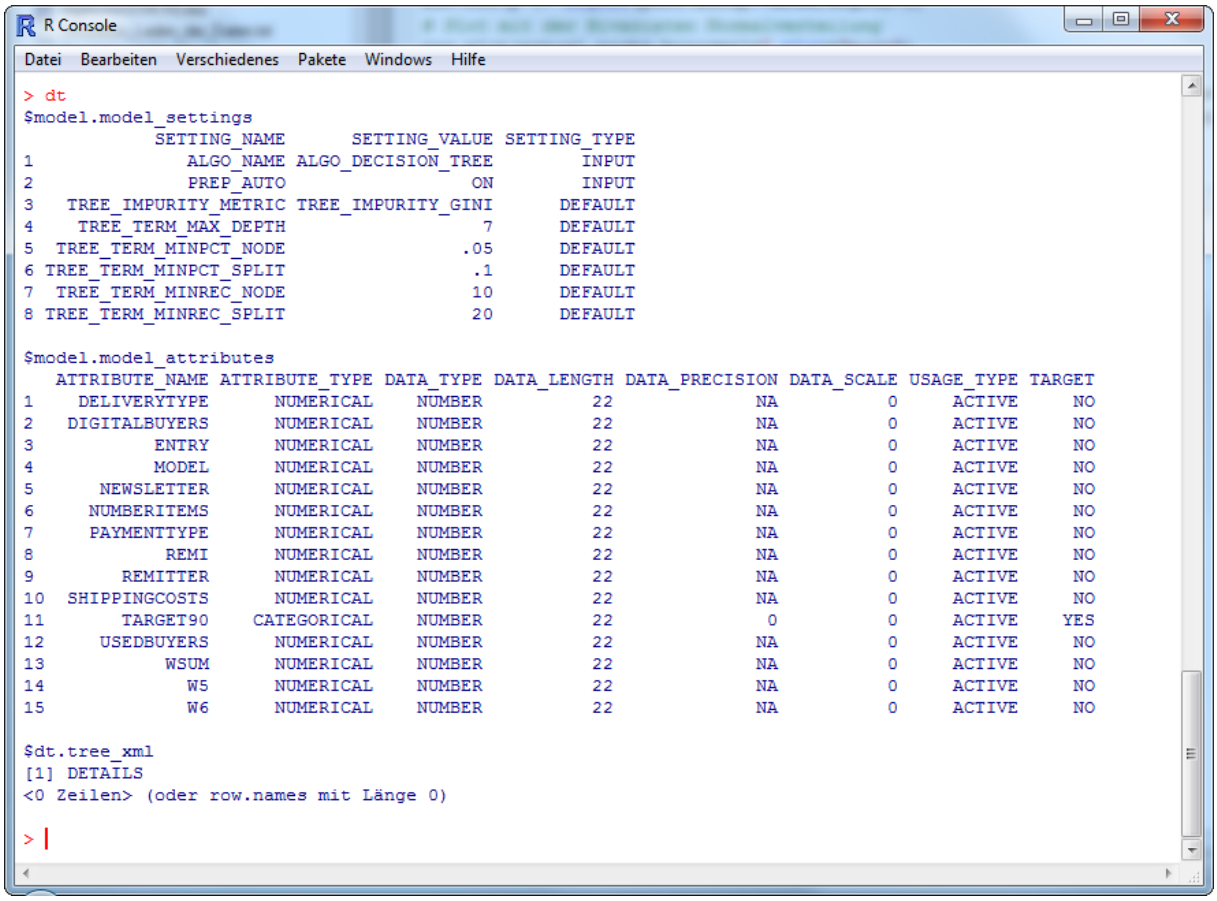

Abbildung 8: Anzeige der Attribute eines Modells

Anwendung und Evaluation des Modells

Die entwickelten Modelle werden bei diesem Schritt angewendet und evaluiert, um deren Qualität zu prüfen. Für den Fall, dass der Modellname nicht mehr bekannt ist, können sämtliche Data Mining Modelle, die im verwendeten Datenbankschema existieren, mit folgendem Befehl aufgelistet werden:

```
modelList <- RODM_list_dbms_models(DB)
```

Ähnliches existiert auch in R selbst und ist von großer Bedeutung. Mit dem Befehl `ls()` werden sämtliche Objekte aufgelistet, welche im aktuellen Workspace vorhanden sind. Das ist notwendig, falls eine Arbeit fortgesetzt werden soll und man sich nicht mehr an die konkreten Objektnamen erinnert.
Die Anwendung eines Modells geschieht mit R-ODM mit Hilfe der Methode `RODM_apply_model()` und ist für sämtliche Algorithmen gültig. Als Parameter sind lediglich eine gültige Datenbankverbindung, die Zieltabelle, der Modellname und Spalten, die im Ergebnis mit angezeigt werden sollen. In unserem Fall sieht der Aufruf wie folgt aus:

```
dtapply <- RODM_apply_model(database=DB, data_table_name="TRAINING_DATA",
model_name="DTMODEL1", supplemental_cols="TARGET90")
```

Hiermit wird anhand des Modells vorhergesagt, ob die Kunden erneute Einkäufe tätigen oder nicht. Da es sich dabei um dieselben Daten handelt wie bei der Erstellung des Modells, wird das Zielattribut im

Ergebnis mit angezeigt und bietet die Möglichkeit, Unterschiede zwischen vorhergesagtem und tatsächlichem Zielwert zu erkennen. Zu beachten ist hierbei, dass `dtapply` kein Data Frame sondern eine Liste ist. Befehle wie `head()` schlagen daher zunächst fehl. Mit Hilfe des Attributs `dtapply$model.apply.results` kann jedoch direkt auf den Data Frame mit dem Ergebnis zugegriffen werden. Wichtig für den Anwender ist nun zu wissen, wie gut das Modell die Werte tatsächlich vorhergesagt hat. Die sogenannte *Confusion Matrix*, welche die konkrete Anzahl von richtig und falsch vorhergesagten Datensätzen anzeigt, lässt sich wie folgt ermitteln. Aus dem Ergebnis werden die Spalten, welche die tatsächlichen und vorhergesagten Werte enthalten, in separaten Vektoren gespeichert und anschließend mit dem Befehl `table()` verknüpft und die Werte (hier ausschließlich 0 oder 1) miteinander verglichen.

```
actual <- dtapply$model.apply.results[,"TARGET90"]
predicted <- dtapply$model.apply.results[,"PREDICTION"]
table(actual,predicted)
```

Darüber hinaus existieren für diesen Anwendungsfall eigenständige Funktionen in entsprechenden Paketen. Zum Beispiel lässt sich mit der Funktion `cmx()` aus dem Paket *PresenceAbsence* die Confusion Matrix direkt anzeigen. Weitere Möglichkeiten zur Bewertung des Data Mining Modells sind das sogenannte *Lift Chart* und die *Receiver Operating Characteristic (ROC) Kurve*. Für beide Grafiken ist es erforderlich, weitere Pakete zu installieren und zu laden (*ROCR* bzw. *Verification*).
Mit Hilfe der folgenden Befehlssequenz lässt sich mit dem Paket *Verification* eine ROC Kurve darstellen. Die Variable probs enthält dabei sämtliche Wahrscheinlichkeiten für die Vorhersage des Wertes 1 des Zielattributs. Die zweite Befehlszeile bewirkt, dass Kennzahlen der ROC Kurve berechnet und in `perf.auc` gespeichert werden. Anschließend wird das Diagramm erstellt. Die beiden letzten Befehlszeilen bewirken, dass wichtige Werte an der konfigurierten Position (Parameter 1 und 2) dem Diagramm hinzugefügt werden. Der Wert *AUC ROC* gibt beispielsweise die Fläche unter der Kurve an und ist ein Indiz für die Qualität des Modells. Das Resultat ist in *Abbildung 9* zu erkennen. Die Diagonale repräsentiert die Trefferquote einer zufälligen Schätzung und die rote Kurve zeigt die Trefferquote des Modells. Damit wird beispielsweise deutlich, dass das hier erstellte Modell noch nicht besonders zuverlässig ist.

```
probs <- dtapply$model.apply.results[,"'1'"]
perf.auc <- roc.area(ifelse(actual=="1",1,0),probs)
roc.plot(actual,probs,binormal=T,plot="both")
text(0.7,0.4,labels=paste("AUC  ROC:",signif(perf.auc$A,digits=3)))
text(0.7,0.3,labels=paste("p-value:",signif(perf.auc$p.value,digits=3)))
```

Ein Lift Chart lässt sich unter Verwendung des Pakets ROCR zum Beispiel mit folgenden Befehlen erstellen.

```
pred  <-  prediction(dtapply$model.apply.results[,"'1'"],dtapply$model.apply
.results[,"TARGET90"])
perf <- performance(pred,"lift","rpp")
plot(perf, main="lift curve", colorize=T)
```

Die hier genannten Möglichkeiten zur Bewertung werden vom Oracle Data Miner zwar automatisch erstellt, R-ODM bietet jedoch den Vorteil, dass verschiedene Visualisierungsmöglichkeiten verwendet werden können.
Sollte das Data Mining Modell nicht den erhofften Erwartungen entsprechen oder nicht mehr weiter benötigt werden, kann dies mit Hilfe des Befehls `RODM_drop_model(DB, "DTMODEL1")` aus der

Datenbank entfernt werden. Eine gewisse Sorgfalt ist zudem ratsam, da der Benutzer sonst schnell den Überblick über die erstellten Modelle verlieren kann.

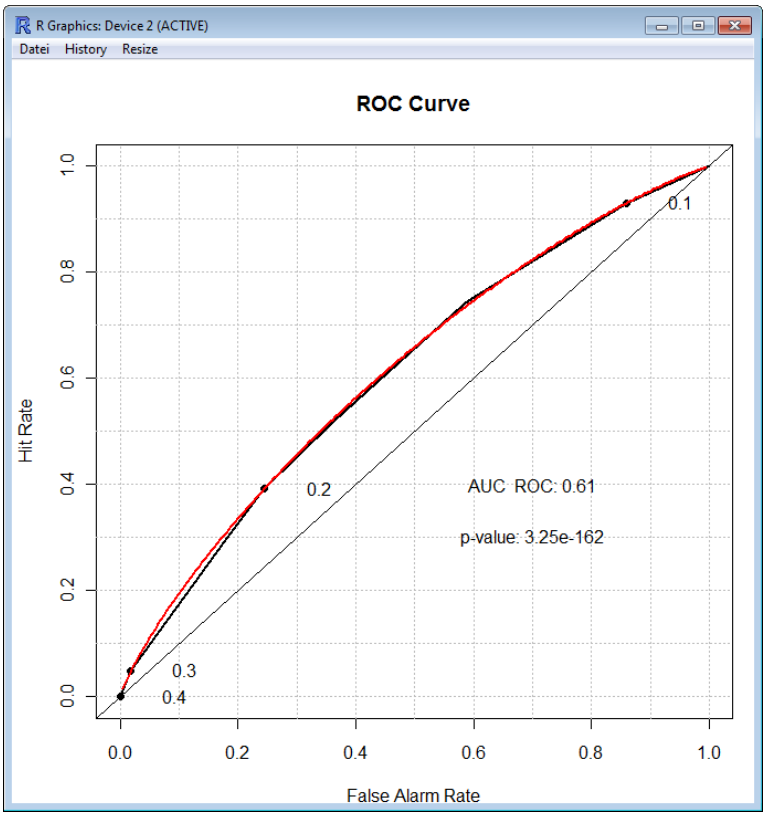

Abbildung 9: Receiver Operating Characteristic - Kurve

Ausblick: Automatisierung des Data Mining Prozesses mit R-Skripten

Um R-ODM auch produktiv einsetzen zu können, besteht die Möglichkeit, den gesamten Vorgang automatisiert ablaufen zu lassen, indem ein entsprechend konfiguriertes R-Skript automatisch ausgeführt wird. Dieses Skript enthält schließlich die Befehle zum Erstellen des Modells, der Anwendung und der Weiterverarbeitung der Ergebnisse. Die produktive Nutzung von Data Mining Ergebnissen kann sehr unterschiedlich ausfallen. Im hier verwendeten Beispiel wäre denkbar, dass die Kundentabelle aktualisiert wird und ein Flag gesetzt wird, das andeutet, ob es sich bei dem Kunden um einen treuen Kunden handelt oder nicht. Um dies zu erreichen ist es erforderlich, sich etwas näher mit dem RODBC Paket und dessen Möglichkeiten zu beschäftigen, da R-ODM lediglich eine Methode bietet, um einen Data Frame als eigenständige Tabelle abzulegen, hier jedoch einzelne Datensätze in einer Tabelle aktualisiert werden sollen. RODBC bietet schließlich eigenständige Methoden, um Tabellen zu kopieren, Daten oder lediglich die Struktur einer Tabelle abzufragen.

Fazit

Die Möglichkeit, die Oracle Data Mining Algorithmen auch in einer R-Umgebung nutzen zu können, scheint gerade für Testzwecke oder die Entwicklung von Prototypen sehr sinnvoll zu sein. Durch den modularen Aufbau von R bietet sich dem Benutzer eine flexible Umgebung, die für die meisten Anwendungsfälle entsprechende Lösungen bereithält. Die Vielzahl der grafischen Oberflächen vereinfacht zudem die Arbeit mit R selbst und somit sind für grundlegende Datenanalysen weder tiefgreifende Datenbank- noch R-Kenntnisse erforderlich. Als Kritikpunkt könnte man anführen, dass

sämtliche Daten in R im Hauptspeicher gehalten werden und somit große Datenmengen nicht verarbeitet werden können, doch hierfür existieren ebenfalls Pakete, welche große Datenmengen aufteilen und teilweise auf die Festplatte auslagern. Die Automatisierung eines Data Mining Prozesses mit Hilfe von R-Skripten gestaltet sich ebenfalls sehr einfach, jedoch bietet hier die Verwendung von Java oder PL/SQL mehr Möglichkeiten und erscheint für eine Produktivumgebung sinnvoller zu sein. Die Stärken von R-ODM liegen hauptsächlich bei den Visualisierungsmöglichkeiten und der flexiblen Handhabung der Daten. Statistische Werte können sehr schnell ermittelt und Ergebnisse weiterverarbeitet werden, ohne beispielsweise auf Berechtigungen für Tabellen oder Ähnliches achten zu müssen. Damit eignet sich R-ODM besonders in der Phase des Data Understanding im Rahmen des Profiling sowie für Demonstrations- und Testzwecke und gewährt auch Anwendern ohne SQL Know-How Zugriff auf die Data Mining Funktionen der Oracle Datenbank.

Kontaktadresse:

Prof. Dr. Reinhold von Schwerin
Hochschule Ulm
Prittwitzstraße, 10
D-89075 Ulm

Telefon: +49 (0) 731-50 28259
E-Mail r.schwerin@hs-ulm.de

Florian Langenbruch
Hochschule Ulm
Prittwitzstraße, 10
D-89075 Ulm

Telefon: +49 (0) 731-50 28244
E-Mail langenbruch@hs-ulm.de

Jörg Bellan
Hochschule Ulm
Prittwitzstraße, 10
D-89075 Ulm

Telefon: +49 (0) 731-50 28249
E-Mail bellan@hs-ulm.de