Statistisches Consulting

Master Studiengang Statistik
Institut für Statistik
Ludwig-Maximilians-Universität München

Online-Marketing der Interhyp AG
Analyse von Tracking-Daten

Daniel Fuckner d.fuckner@gmx.de
Markus Vogler markus@vogler-lindau.de

Projektpartner:
Interhyp AG

Betreuer:
Dr. Fabian Scheipl

München, 11.11.2011

Abstract

Die Interhyp AG ist Vermittler für private Baufinanzierungen. Das primäre Ziel des Marketing der Interhyp AG ist die Kundenakquise. Da etwa 80% aller Kundenanträge online abgeschickt werden, liegt der Fokus auf dem Online-Marketing, das über verschiedene Kampagnen verfügt. Beispiele sind Kooperationen mit anderen Unternehmen, bezahlte Anzeigen bei Suchmaschinen oder Bannerschaltungen.
Die Refined Labs GmbH ist verantwortlich für das Online-Tracking der Werbekampagnen der Interhyp AG. Durch Online-Tracking werden die Werbekontakte eines potentiellen Kunden zu einem sogenannten Funnel zusammengefasst. Am Ende eines jeden Funnels steht das Ausfüllen eines Onlineantrages oder der Abbruch. Man spricht von konvertierten beziehungsweise nicht-konvertierten Funnels.
In dieser Arbeit werden die Daten zunächst anhand deskriptiver Analysen vorgestellt. Außerdem werden Methodik und Ergebnisse eines Sequential Pattern Mining-Algorithmus sowie eines zeitdiskreten Survival-Modells, welches mittels Stochastic Gradient Boosting geschätzt wurde, beschrieben. Zudem wurden die konvertierten und nicht-konvertierten Funnels in Form eines Netzwerkes visualisiert. Die Daten für dieses Netzwerk und ein Programm, dass die interaktive Betrachtung ermöglicht, ist im elektronischen Anhang enthalten.

Inhaltsverzeichnis

1 Einleitung

Die Interhyp AG ist Vermittler für private Baufinanzierungen. Das heißt, sie wählt aus einem Angebot von verschiedenen Darlehensgebern die optimale Finanzierungsstruktur für einen Kunden aus. Das Unternehmen wurde 1999 von den ehemaligen Goldman-Sachs-Bankern Robert Haselsteiner und Marcus Wolsdorf gegründet. Sechs Jahre später eröffnete die Interhyp AG erste Niederlassungen und konnte gleichzeitig den erfolgreichsten deutschen Börsengang des Jahres verzeichnen. Nach weiteren drei Jahren erfolgte die Übernahme durch ING DIRECT, der weltweilt größten und erfolgreichsten Direktbanken-Gruppe. Heute ist die Interhyp AG der größte Vermittler für private Baufinanzierungen in Deutschland, wurde acht mal in Folge als "Bester Baufinananzierer"(Zeitschrift €, Ausgabe 08/2013) ausgezeichnet und verfügt über mehr als 60 Beratungsstandorte mit über 1.000 Mitarbeitern.
Das primäre Ziel des Marketing der Interhyp AG ist die Kundenakquise. Da etwa 80% aller Kundenanträge online abgeschickt werden, liegt der Fokus der Marketing-Abteilung auf dem Online-Marketing, das über verschiedene Kanäle verfügt. Beispiele sind Kooperationen mit anderen Unternehmen wie Immobilienscout24, bezahlte Anzeigen bei Suchmaschinen, Newsletter oder diverse Bannerschaltungen. Durch Online-Tracking können die Werbekontakte eines potentiellen Kunden mit der Interhyp AG zusammengefasst werden. So entsteht ein sogenannter Funnel, wie es in Abbildung 1 skizziert ist. Jeder potentielle Kunde hat einen oder mehrere aufeinanderfolgende Kontaktpunkte, wobei jeder Kontaktpunkt die genaue Zeit des Kontaktes sowie die Art des Kontaktes, das heißt die Information über welche Kampagne es zu dem Kontakt kam, enthält. Am Ende eines jeden Funnels steht der Abbruch der Beobachtung oder im Idealfall das Ausfüllen eines Onlineantrages durch den Kunden.

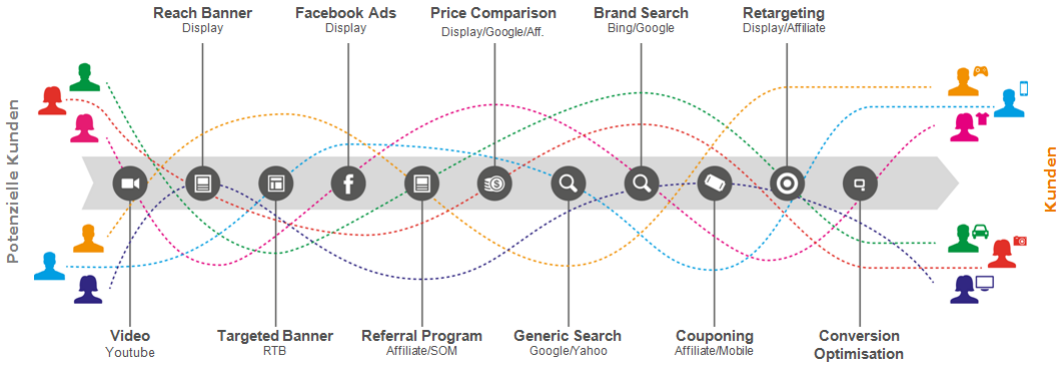

Abbildung 1: Entstehung eines Funnels

Die Refined Labs GmbH ist auf dem Gebiet des Online-Marketing spezialisiert und verantwortlich für das Online-Tracking der Werbekampagnen der Interhyp AG. Ein Funnel beginnt mit dem ersten Online-Werbekontakt eines potentiellen Kunden mit der Interhyp AG und der damit einhergehenden Erstellung eines Cookies. So können alle weiteren Werbekontakte dem potentiellen Kunden eindeutig zugewiesen werden. Das

Tracking endet sobald der potentielle Kunde einen Onlineantrag versendet und damit zum Kunden wird. In diesem Fall spricht man von einem konvertierten Funnel. Wird innerhalb von 90 Tagen kein Onlineantrag versendet, so wird das Cookie nicht weiter verfolgt und der Funnel wird als nicht-konvertiert bezeichnet.
Die Interhyp AG ist primär daran interessiert, ob man dem Abbruch eines Funnels entgegen wirken kann und damit allgemein an Unterschieden zwischen konvertierten und nicht-konvertierten Funnels. Um dieser Fragestellung gerecht zu werden, wurde ein zeitdiskretes Survival-Modell mittels Stochastic Gradient Boosting geschätzt und ein Sequential Pattern Mining-Algorithmus angewendet. Außerdem wurden die Funnels anhand eines Netzwerkes visualisiert.
Die Arbeit ist wie folgt gegliedert. In Kapitel 2 und 3 werden die Datenaufbereitung und die Variablen erklärt. Daraufhin wird die Methodik des Survival-Modells in Kapitel 4, der Sequential Pattern Mining-Algorithmus in Kapitel 5 und das Netzwerk in Kapitel 6 erläutert. Die Ergebnisse dieser Methoden werden in Kapitel 7 vorgestellt und abschließend erfolgt eine Zusammenfassung in Kapitel 8 und die Beschreibung des elektronischen Anhangs in Kapitel 9.

2 Datenlage

Die Daten wurden von der Refined Labs GmbH als SQL-Dump bereitgestellt, der eine Größe von circa 13 Gigabyte hat. Die MySQL-Datenbank enthält die vier Tabellen *project_out*, *redirects_short*, *searchFunnel* und *stage2_transactionHandling*. Mit Hilfe der vorhanden Informationen in *searchFunnel* und *stage2_transactionHandling* konnten die Kontaktpunkte in *redirects_short* in konvertierte und nicht-konvertierte Funnels unterteilt werden. In *projects_out* sind die Kampagnen in Form einer Baumstruktur organisiert. In Absprache mit der Interhyp AG wurden 17 Kategorien ausgewählt, die sich auf den ersten drei Ebenen dieser Baumstruktur befinden. Anhand von IDs wurde jedem Kontaktpunkt eine dieser Kategorien zugewiesen. Die 17 Kampagnen und weitere Features, die aus den Daten erzeugt wurden, werden in Kapitel 3 näher erläutert.
Ein Kontaktpunkt ist entweder ein *Click* oder ein *View*. Man spricht von einem *Click*, wenn der potentielle Kunde tatsächlich etwas angeklickt hat, wobei die genaue Definition von der Kampagne abhängt. Ein *View* wird getrackt, wenn ein Banner oder ähnliches lediglich gesehen, aber nicht angeklickt wird. An dieser Stelle wirft die Datenerhebung allerdings ein Problem für die statistischen Analysen auf. Die *Views* werden für alle konvertierten Funnels gespeichert, für die nicht-konvertierten Funnels allerdings nur, wenn diese bei einem anderen Kunden der Refined Labs GmbH konvertieren. Dass heißt, es ist eine systematische Veränderung der Daten gegeben. Deshalb besteht keine Möglichkeit die *Views* in statistische Analysen, die konvertierte und nicht-konvertierte Funnels vergleichen, einzubeziehen. Die *Views* werden lediglich in Kapitel 3 in einigen Plots betrachtet, die nur konvertierte Funnels enthalten, und von den weiteren Analysen ausgeschlossen.
Nach der Vorverarbeitung der Daten liegen ??? konvertierte und ??? nicht-konvertierte Funnels vor, die nur *Clicks* enthalten. Eine nähere Beschreibung der erstellten Features

erfolgt in Kapitel 3.

3 Deskriptive Analyse

Tabelle 1 enthält eine Übersicht mit den erzeugten Variablen. Diese werden in diesem Kapitel näher betrachtet. Als Position wird im folgenden die Nummer eines Kontaktpunktes innerhalb eines Funnels bezeichnet. Das heißt für jeden Funnel nimmt der erste Kontakt die Position 1 an.

$clickCount \in \mathbb{N}$	Anzahl an *Clicks* bis zur aktuellen Position
$hasClicked \in \{0, 1\}$	Dummyvariable, die angibt ob vor der aktuellen Position schon geklicked wurde (1) oder nicht (0).
$campaign$	Kampagne der aktuellen Position
$campaignLast$	Kampagne der vorherigen Position
$campaignLast2$	Kampagne der vorletzten Position
$weekday \in \{Montag, ...\}$	Wochentag des Kontaktes
$hour \in \{0, 1, \ldots, 23\}$	Uhrzeit des Kontaktes
$timeSinceLast \in \mathbb{R}$	Zeitdifferenz zwischen aktueller und vorheriger Position
$timeSinceFirst \in \mathbb{R}$	Zeitdifferenz zwischen aktueller und erster Position
$freq \in \mathbb{R}$	Frequenz der Kontaktpunkte in einem Funnel

Tabelle 1: Variablenbeschreibung

3.1 Views in den konvertierten Funnels

clickCount

Aufgrund der in Kapitel 2 beschriebenen Problematik bei der Datenerhebung der *Views* können die Variablen *clickCount* und *hasClicked* nur in den konvertierten Funnels betrachtet werden. In Abbildung 2 ist der *clickCount* dargestellt. Auf der x-Achse ist die Position aufgetragen und auf der y-Achse der *clickCount*, das heißt die Häufigkeit der *Clicks* gemittelt über alle konvertierten Funnels für jede Position. Die rote Diagonale wäre erreicht, wenn die konvertierten Funnels nur aus *Clicks* bestehen würden. Es ist zu erkennen, dass die Linie mit der Position ansteigt. An Position 100 ist die mittlere Anzahl der Clicks ???, das heißt im Mittel bestehen die ersten 100 Kontakte eines Funnels aus ??? *Clicks* und 1-??? *Views*. Die Anzahl der *Views* übersteigt die Anzahl der *Clicks* also deutlich.

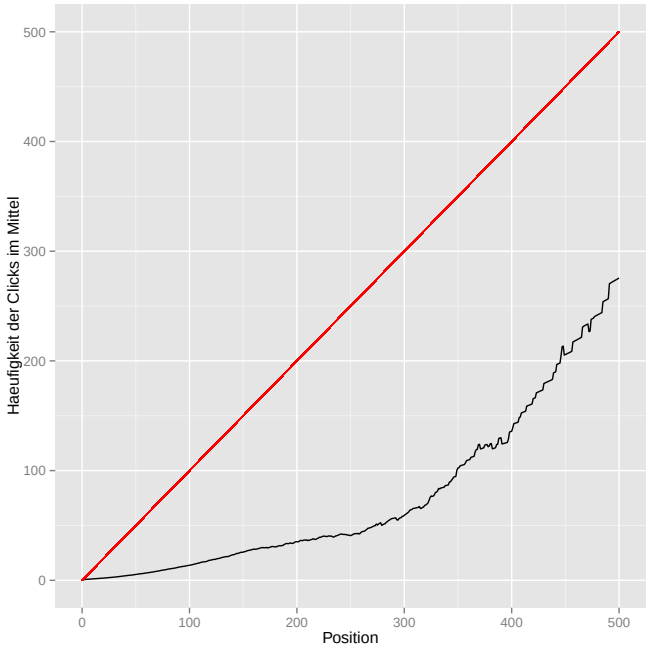

Abbildung 2: Häufigkeit der Clicks im Mittel für jede Position

hasClicked

Die Variable *hasClicked* (siehe Abbildung 3) gibt für jede Position den Anteil der Funnels an, die bis dorthin mindestens einen *Click* enthalten. Dieser Wert nimmt zwischen Position 1 und Position 7 ab. Dies ist dadurch zu erklären, dass es viele Funnels gibt, die *Clicks* enthalten und deren Länge kleiner als 6 ist. Sobald diese Funnels beendet sind, werden sie an der nächsten Position selbstverständlich nicht mehr berücksichtigt. Ab Position 7 steigt die Kurve dann bist zur 1 an. Ein Wert von 1 bedeutet, dass alle Funnels bereits einen *Click* hatten.

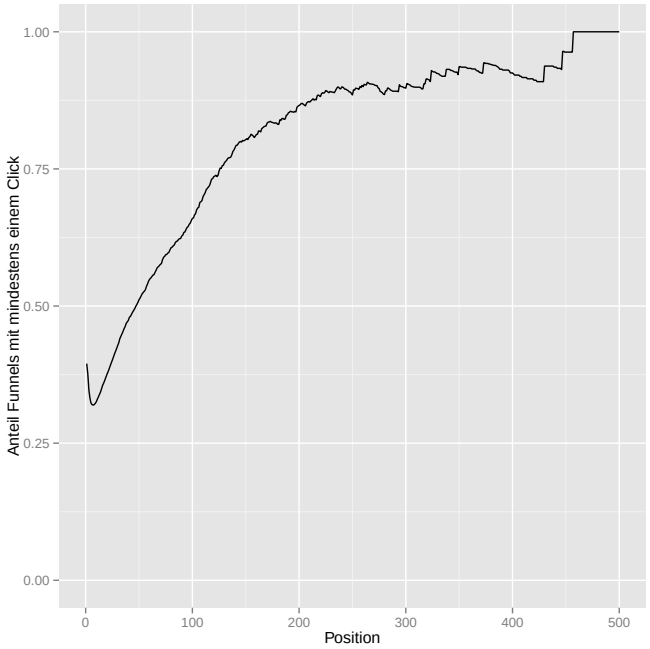

Abbildung 3: Anteil der Funnels mit mindestems einem Click für jede Position

campaign

Nun sollen noch die verschiedenen Kategorien des Features *campagin*, das heißt die verschiedenen Werbeformen betrachtet werden. Tabelle 2 enthält Erklärungen der 17 verwendeten Kategorien.

Kampagne	**Beschreibung**
Affiliate - Partnerprogramm	Partner, die von der Interhyp AG bereitgestellte Werbemittel wie Rechner, Logo oder Banner einbinden
Affiliate - Rest	Partner, die einen Zinsvergleich bereitstellen, welcher das Zinsangebot der Interhyp AG mit deren Wettbewerbern im Vergleich darstellt
Direct	Potentieller Kunde gibt im Browser direkt *www.interhyp.de* ein
Display	Bannerschaltungen
E-Mailing	Mails an Interessenten, die schon einen Antrag gestellt oder ein Infopaket angefordert hatten
Generic	Potentieller Kunde kommt über unbezahlten Link zur Interhyp AG
Kooperationen - Focus Kooperationen - Immonet Kooperationen - Immoscout24 Kooperationen - Immowelt Kooperationen - Rest	Individuelle Zusammenarbeiten mit größeren Partnern, die je nach Vertrag verschiedene Werbemittel auf ihrer Seite einbinden
Newsletter	Regelmäßige Rundschreiben
SEM - Brand	Bezahlte Suchergebnisse, wobei nach *Interhyp* oder ähnlichem gesucht wurde
SEM - Remarketing	Bezahlte Suchergebnisse, wobei der potentielle Kunde bereits zuvor auf der Seite der Interhyp AG war
SEM - Generisch	Bezahlte Suchergebnisse, wobei nach *Baufinanzierung* oder ähnlichem gesucht wurde
SEO	Unbezahlte Suchergebnisse
Social Media	Werbung, vor allem auf *facebook* und *gutefrage.net*

Tabelle 2: Beschreibung der Kampagnen

Abbildung 4 enthält die Verteilung dieser 17 Kategorien in den konvertierten Funnels, das heißt auf der x-Achse ist die relative Häufigkeit aufgetragen und auf der y-Achse die Kategorien. Die blauen Balken repräsentieren die Verteilung in den konvertierten Funnels nur mit *Clicks*, wie sie auch in den späteren Analysen verwendet werden. Die roten Balken enthalten *Clicks* und *Views*.
Es ist zu erkennen, dass die Kampagne *Display* bei den Funnels mit *Views* 84% der gesamten Kontaktpunkte ausmacht. Das heißt die Bannerschaltungen überwiegen deutlich und ansonsten hat nur *Direct* einen Anteil von über 5%.
Werden die *Views* nicht berücksichtigt so verteilen sich die Kampagnen besser. *Display* macht jetzt weniger als 10% aus und *Direct* ist mit über 35% jetzt die am häufigsten auftretende Kampagne. Außerdem haben auch *SEO*, *SEM - Generisch*, *SEM - Brand*, *Kooperationen - Immoscout24* und *Affiliate - Partnerprogramm* einen Anteil von über

5%. Die restlichen Kampagnen, besonders *Social Media*, machen nur einen kleinen Teil der Daten aus.

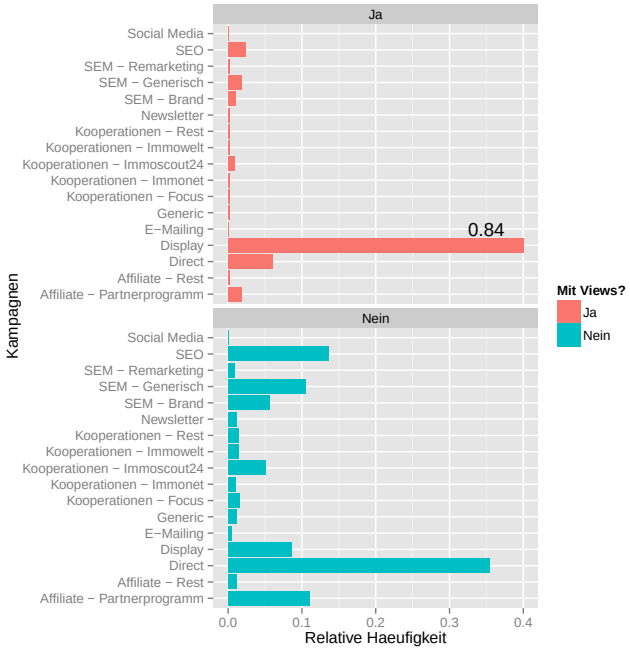

Abbildung 4: Kampagnen der konvertierten Funnels mit und ohne Views

3.2 Vergleich von konvertierten und nicht-konvertierten Funnels

Nachdem bis hierhin nur die konvertierten Funnels mit Augenmerk auf den *Views* betrachtet wurden, sollen in diesem Kapitel die konvertierten und nicht-konvertierten Funnels miteinander verglichen werden. Das heißt, die *Views* werden von nun an nicht mehr in die Analysen mit einbezogen.

weekday

Die Variable *Weekday* gibt an, an welchem Wochentag ein Konaktpunkt aufgetreten ist. Abbildung 5 enthält diesbezüglich Histogramme. Die roten Balken entsprechen den konvertierten und die blauen Balken den nicht-konvertierten Funnels. Für weitere Plots in diesem Kapitel gelten die selben Farben. Außerdem ergeben die blauen und roten Balken erneut aufsummiert jeweils eins, das heißt sie spiegeln die Verteilung wieder.
Es ist zu erkennen, dass die Häufigkeit der Kontaktpunkte von Montag bis Samstag sinkt. Dieser Trend ist in den konvertierten Funnels etwas stärker. Dort sinkt die relative Häufigkeit von ??? am Montag auf ??? am Samstag. Bei den nicht-konvertierten Funnels sinkt die relative Häufigkeit von ??? am Montag auf ??? am Samstag. Außerdem ist der Sonntag bei den nicht-konvertierten mit Abstand der stärkste Tag, während in den konvertierten Funnels der Montag etwas stärker ist als der Sonntag.

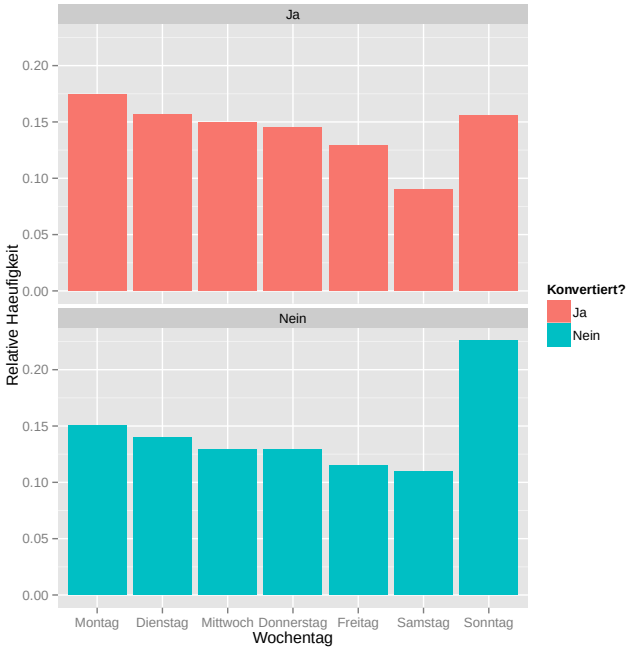

Abbildung 5: Wochentage der Kontaktpunkte

hour

Analog zu der Abbildung mit den Wochentagen, enthält Abbildung 6 Informationen zu der Uhrzeit der Kontaktpunkte in konvertierten und nicht-konvertierten Funnels. Hierfür wurden die Minutenangaben der Uhrzeit jeweils abgeschnitten, so dass sich die Beobachtungen $0, 1, ..., 23$ ergeben.
Die Verteilungen in konvertierten und nicht-konvertierten Funnels sind sich sehr ähnlich, so dass keine deutlichen Unterschiede erkennbar sind. Insgesamt kann man zusammenfassen, dass in der Nacht zwischen zwei und sechs Uhr sehr wenige Kontaktpunkte stattfinden. Ab sechs Uhr steigt die relative Häufigkeit der Kontaktpunkte bis circa elf Uhr an und dann bleibt sie konstant bis circa 21 Uhr. Daraufhin fällt die relative Häufigkeit wieder ab. Dies ist lediglich darauf zurück zu führen, dass nachts weniger Menschen online sind.

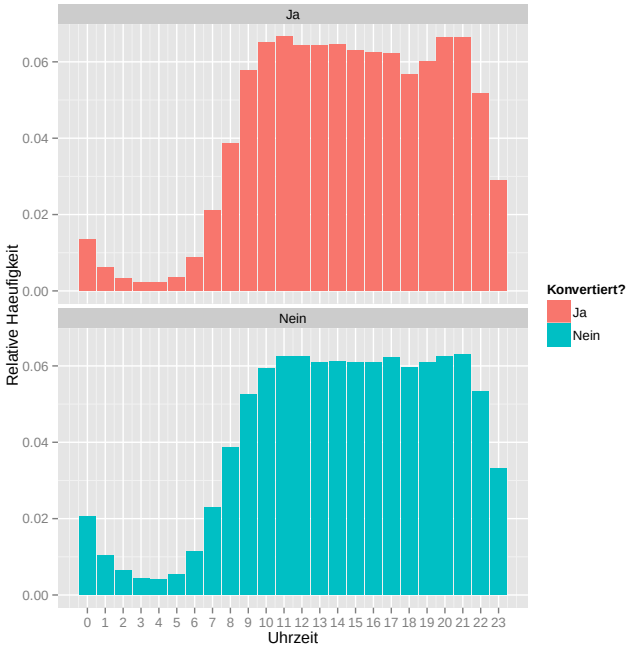

Abbildung 6: Uhrzeit der Kontaktpunkte

campaign

Wie bereits in Kapitel 3.1 werden die verschiedenen Kampagnen betrachtet. Allerdings werden an dieser Stelle die konvertierten und nicht-konvertierten Funnels miteinander verglichen, wobei die *Views* nicht berücksichtigt werden (siehe Abbildung 7. Das heißt, die Verteilung der konvertierten Funnels ohne *Views* aus Kapitel 3.1 entspricht der Verteilung der konvertierten Funnels in diesem Abschnitt. Eine nähere Beschreibung der unterschiedlichen Kampagnen ist in Abbildung 2 zu finden.
Die Verteilung in den konvertierten Funnels wurde bereits in Kapitel 3.1 etwas näher beleuchtet. Während dort *Direct* mit Abstand die stärkste Kampagne ist, sind in den nicht-konvertierten Funnels *Affiliate - Partnerprogramm* und *Display* die stärksten Kategorien und *Direct* ist lediglich drittstärkste mit einer relativen Häufigkeit von ???. Wie in den konvertierten Funnels haben hier auch *SEO*, *SEM - Generisch* und *Kooperationen - Immoscout24* einen Anteil von über 5%. *SEM - Brand* tritt in den konvertierten Funnels deutlich häufiger auf als in den nicht-konvertierten. Für die restlichen Kampagnen liegen insgesamt wenige Daten vor.

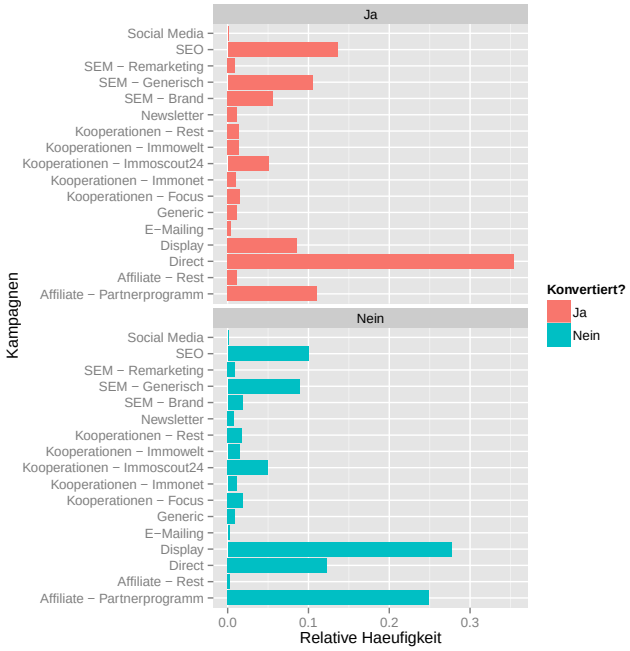

Abbildung 7: Kampagnen der konvertierten und nicht-konvertierten Funnels

funnelLength

Die *funnelLength* gibt die Anzahl der Kontaktpunkte eines Funnels an. In Abbildung 8 sind nur diejenigen Funnels dargestellt, deren Länge 20 Kontaktpunkte nicht überschreitet, da es, relativ gesehen, sehr wenige längere Funnels gibt.
Von den nicht-konvertierten Funnels haben 75% nur einen Kontaktpunkt. Von dort nimmt die relative Häufigkeit der Funnels mit steigender Länge sehr schnell ab. Von den konvertierten Funnels haben 41% nur einen Kontaktpunkt. Das heißt, relativ betrachtet, gibt es dort mehr Funnels mit mehreren Kontaktpunkten. Allerdings gibt es insgesamt deutlich mehr nicht-konvertierte als konvertierte Funnels, so dass die absoluten Anzahl für die nicht-konvertierten stets größer ist. Der Mittelwert beziehungsweise der Median der Länge der Funnels ist bei den konvertierten Funnels 57 beziehungsweise 8 und bei den nicht-konvertierten 8 beziehungsweise 2. (**hast du hier auch Last oder so gleich 1 gesetzt bei der Berechnung von mean und med?**)

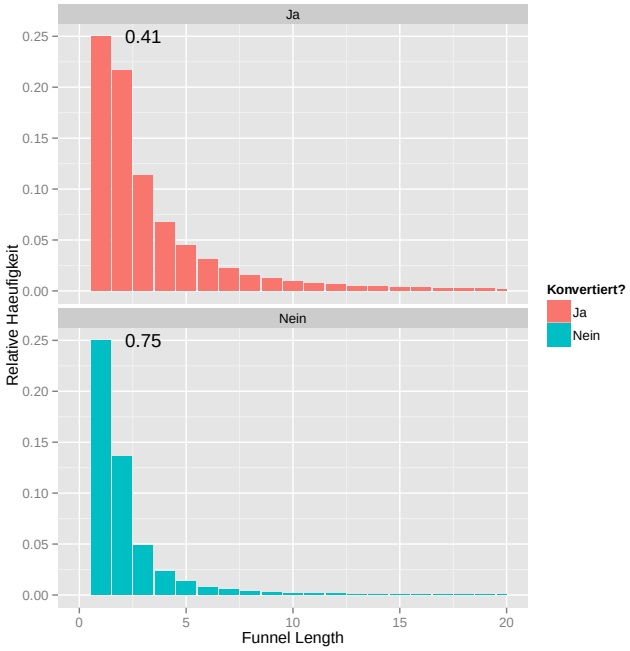

Abbildung 8: Länge der Funnels in konvertierten und nicht-konvertierten Funnels

timeSinceFirst

Das Feature *timeSinceFirst* gibt für jede Position die verstrichene Zeit seit dem ersten Kontaktpunkt an. In Abbildung 9 wird dieses nur für die letzte Position der Funnels geplottet, so dass die Gesamt-Beobachtungsdauer der Funnels betrachtet wird, das heißt die verstrichene Zeit zwischen dem ersten und dem letzten Kontaktpunkt eines jeden Funnels. Auf der x-Achse ist die Beobachtungs dauer in Tagen von 0 bis 50 aufgetragen. Hier gestaltet sich ein ähnliches Bild wie in Abbildung 8. Von den nicht-konvertierten Funnels haben 58% eine Beobachtungsdauer von weniger als einem Tag. Längere Beobachtungsdauern treten deutlich seltener auf. Von den konvertierten Funnels haben 33% eine Beobachtungsdauer von weniger als einem Tag und längere Beobachtungsdauern treten, relativ betrachtet, häufiger auf als in den nicht-konvertierten Funnels.
Auffällig ist außerdem, dass die relative Häufigkeit im Abstand von sieben Tagen wieder leicht zunimmt und ansonsten fällt. Dies ist darauf zurückzuführen, dass Sonntag und Montag die zwei Tage mit den häufigsten Kontakten sind und beispielsweise Bannerschaltungen dort besonders häufig eingesetzt werden. Bei den konvertierten Funnels liegt der Mittelwert der Beobachtungsdauer bei 21.1 Tagen und bei den nicht-konvertierten Funnels bei 5.3 Tagen.

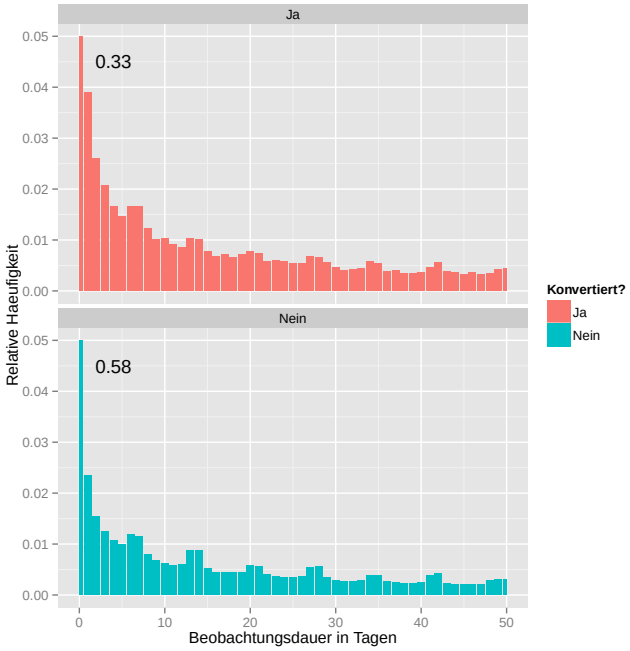

Abbildung 9: Beobachtungsdauer in Tagen der konvertierten und nicht-konvertierten Funnels

timeSinceLast

Die Variable *timeSinceLast* gibt die verstrichene Zeit zweier aufeinander folgender Kontaktpunkte an. Diese ist in Abbildung *timeSinceLast* abgebildet, wobei nun alle Kontaktpunkte berücksichtigt werden und nicht nur der letzte, wie es bei *timeSinceFirst* der Fall war.
Die relative Häufigkeit der Abstände, die kürzer als ein Tag sind ist bei den nicht-konvertierten Funnels höher als bei den konvertierten. Ansonsten sind die Werte bei den konvertierten Funnels höher, wobei wieder ein Abfall mit der Zeit und eine wöchentlich Periodizität zu erkennen sind.
Da hier nur die Verteilungen jeweils innerhalb der konvertierten und nicht-konvertierten Funnels verglichen werden, ist *timeSinceLast* keine geeignetes Maß zum Vergleich der Frequenzen der Kontaktpunkte. Dafür wurde die Variable *freq* erzeugt, die im nächsten Abschnitt beschrieben wird.

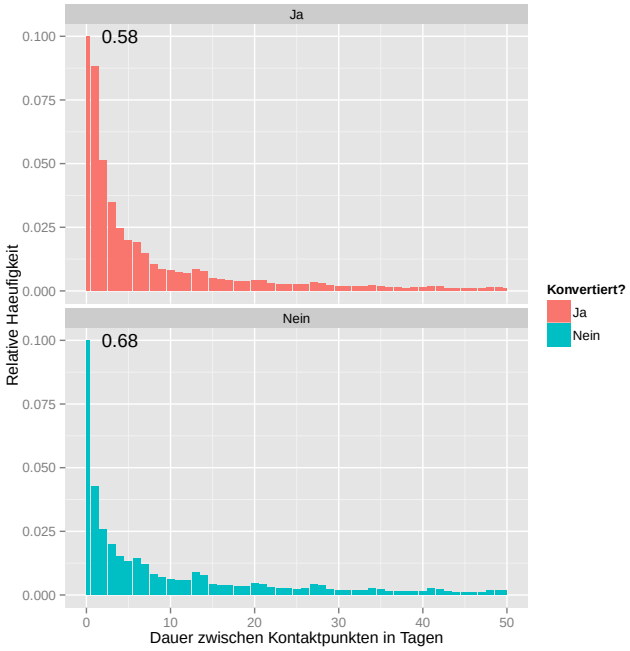

Abbildung 10: Dauer zwischen zwei Kontaktpunkten in den konvertierten und nicht-konvertierten Funnels

freq

Die Frequenz wird wie folgt berechnet. Die Daten werden dahingehend gefiltert, dass für jeden Funnel nur der letzte Kontaktpunkt vorhanden ist. Die Variable *timeSinceFirst* gibt somit wieder die Beobachtungsdauer des gesamten Funnels an. Daraufhin wird die Länge des Funnels (siehe Abbildung 8) durch die Beobachtungsdauer in Stunden geteilt, so dass eine Größe ensteht, die angibt, wieviel Konaktpunkte der jeweilige Funnel pro Stunde hatte. Diese Frequenz wird in Abbildung 11 abgebildet, wobei auf der x-Achse die Länge der Funnels zwischen vier und 25 aufgetragen. Für Funnels mit einem Kontaktpunkt existiert offentsichtlich keine Frequenz und die Längen zwei und drei werden nicht mit abgebildet, da die Frequenz dort vergleichsmäßig groß ist. Außerdem sind einige Boxplots nach oben hin abgeschnitten, da die Grenze der y-Achse auf 0.15 gesetzt wurde, damit die Boxplots besser sichtbar sind. Die roten Boxplots entsprechen den Frequenzen der konvertierten und die blauen den Frequenzen der nicht-konvertierten Funnels.
Für die Funnel Länge zwei liegt der Median der Frequenzen bei ??? in den konvertierten und bei ??? in den nicht-konvertierten Funnels sowie für die Länge drei bei ??? beziehungsweise ???.
Insgesamt ist zu erkennen, dass die Frequenzen in den nicht-konvertierten Funnels höher zu sein scheint. Das heißt in denjenigen Funnels die zu keiner Konvertierung führen liegen die Kontaktpunkte näher beieinander, während sie in den konvertierten Funnels mehr über die Zeit verteilt sind. Daraus könnte man schließen, dass in den nicht-konvertierten Funnels eine Art Übersättigung eintritt, da sie mit zu viel Werbung konfrontiert werden.

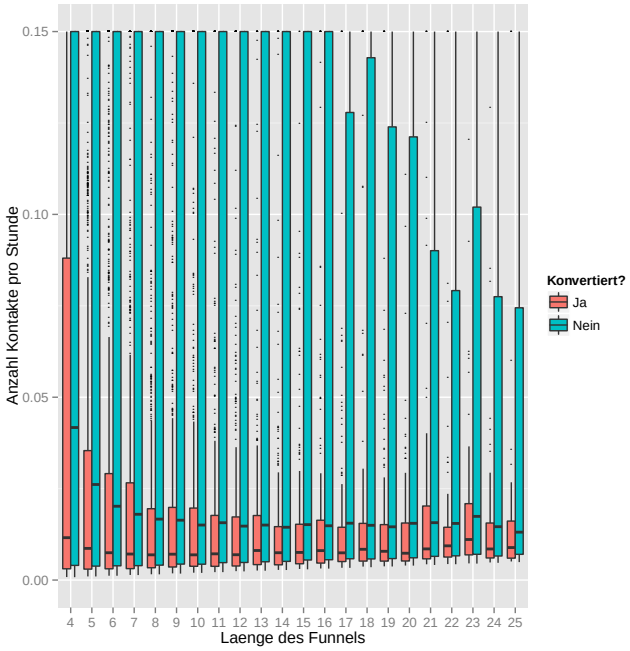

Abbildung 11: Frequenz der Kontaktpunkte in konvertierten und nicht-konvertierten Funnels

4 Zeitdiskretes Survival-Modell

4.1 Lebensdauer-Modell

Aufgrund der in Kapitel 2 beschriebenen Datenlage erscheint die Anwendung eines Modells aus dem Feld der Lebensdaueranalyse intuitiv. Es wird die Zeit bis zu einem Ereignis betrachtet, welches in diesem Fall das Ausfüllen eines Online-Antrages ist. Der Kunde befindet sich während der Beobachtungsspanne im transienten Zustand bis er durch die Konvertierung in den absorbierenden Zustand wechselt, an dem die Beobachtung endet. Tritt am Ende der Beobachtung eines Kunden keine Konvertierung ein, so spricht man von einer Rechtszensierung.
Die Position gibt die Nummer des Kontaktpunktes an und bildet die Zeitachse des Modells. Das heißt, es handelt sich um ein zeitdiskretes Modell und die Zielvariable y_{ip} (1) nimmt den Wert Eins an, wenn Kunde i an der Position p konvertiert ist. Für alle vorherigen Positionen eines konvertierten Funnels und für alle Positionen eines Nichtkonvertierten Funnels nimmt y_{ip} den Wert Null an. N_p ist die Anzahl der Beobachtungen an Position p. Diese nimmt mit steigendem p ab, da in jeder Position Funnels konvertieren oder Beobachtungen ohne Konvertierung enden. Deshalb wird das Modell nur auf die ersten 25 Positionen angewendet, da für spätere Positionen nicht ausreichend konvertierte Funnels vorliegen.

$$y_{ip} = \begin{cases} 1 & \text{Beobachtung } i \text{ konvertiert an Position } p \\ 0 & \text{sonst} \end{cases}, \; p = 1, ..., 25, \; i = 1, ..., N_p \tag{1}$$

Das Modell schätzt die Hazardrate λ_{ip} (2), das heißt die Wahrscheinlichkeit, dass Beobachtung i an Position p konvertiert unter der Bedingung, dass die Länge des Funnels von Beobachtung i größer oder gleich p ist, was lediglich bedeutet, dass für Beobachtung i an Position p überhaupt noch ein Kontaktpunkt vorliegt. Außerdem wird auf die Features x_{ip} bedingt, die später noch näher erläutert werden.

$$\lambda_{ip} = P(y_{ip} = 1 | funnelLength_i \geq p, x_{ip}) \tag{2}$$

Die Hazardrate wird mittels eines Logit-Modells (3-4) mit der Zielvariable y_{ip} an jeder Position p seperat geschätzt. Die Annahmen des Modells sind, dass die $y_{ip}|x_{ip}$ unabhängig Bernoulli-verteilt sind mit der Hazardrate λ_{ip} als Parameter und der Erwartungswert wird anhand der Responsefunktion h mit der Prädiktorfunktion f_{ip} verknüpft.

$$y_{ip}|x_{ip} \overset{ind}{\sim} Bin(1, \lambda_{ip}) \tag{3}$$

$$E(y_{ip}|x_{ip}) = P(y_{ip} = 1|x_{ip}) = \lambda_{ip} = h(f_{ip}) = \frac{\exp(f_{ip})}{1 + \exp(f_{ip})} \tag{4}$$

Aus diesen Annahmen lässt sich die Likelihood (5) und die Log-Likelihood (6) des Modells ableiten.

$$L(\lambda_{ip}) = \prod_{i=1}^{N_p} \lambda_{ip}^{y_{ip}} (1 - \lambda_{ip})^{1 - y_{ip}} \tag{5}$$

$$\begin{aligned} l(\lambda_{ip}) = \ln(L(\lambda_{ip})) &= \sum_{i=1}^{N_p} (y_{ip} \ln(\lambda_{ip}) + (1 - y_{ip}) \ln(1 - \lambda_{ip})) \\ &= \sum_{i=1}^{N_p} (y_{ip} f(x_{ip}) - \ln(1 + \exp(f(x_{ip})))) \end{aligned} \tag{6}$$

Damit ergibt sich der binomielle Verlust aus der negativen Log-Likelihood (7) und das Logit-Modell ist lösbar durch die Minimierung dieses Verlusts (8).

$$L(y, f) = -yf + \ln(1 + \exp(f)) \tag{7}$$

$$\arg\min_{\beta_p} \sum_{i=1}^{N_p} L(y_{ip}, f(x_{ip})) \tag{8}$$

Um ein gutes Prognose-Modell zu entwickeln, wird eine Ensemble-Methode angewendet, die im nächsten Abschnitt vorgestellt wird.

4.2 Stochastic Gradient Boosting

Algorithmus

Stochastic Gradient Boosting ist eine Ensemble-Methoden, die durch mehrfache Anwendung des sogenannten Basis-Lerners ein Ensemble von Schätzern für eine Prognosefunktion liefert. Durch Aggregation der Schätzer erhält man die endgültige Prognosefunktion. Ein sehr beliebter Basis-Lerner sind Stümpfe, das heißt Bäume mit nur einem Split. Einige Vorteile von Bäumen sind, dass sie mit kategoriellen Features, Ausreißern und fehlenden Werten umgehen können. Außerdem wird der schwachen Prognoseleistung von Bäumen durch die Kombination mit Boosting entgegen gewirkt.
Gesucht ist also eine Prognosefunktion, die den Erwartungswert einer Verlustfunktion minimiert. Als Verlustfunktion wird der binomielle Verlust (7) verwendet, wobei sich die Prädiktorfunktion wie folgt ergibt.

$$\begin{aligned} f(x_{ip}) =& \text{offset}(\hat{\lambda}_{i,p-1}) + \\ & f_{weekday,p}(\text{weekday}_{ip}) + \\ & f_{hour,p}(\text{hour}_{ip}) + \\ & f_{campaign,p}(\text{campaign}_{ip}) + \\ & f_{campaignLast,p}(\text{campaign}_{i,p-1}) + \\ & f_{campaignLast2,p}(\text{campaign}_{i,p-2}) + \\ & f_{timeSinceLast,p}(\text{timeSinceLast}_{ip}) + \\ & f_{timeSinceFirst,p}(\text{timeSinceFirst}_{ip}) \end{aligned} \tag{9}$$

Der Prädiktor ist also eine additive Funktion von Treppenfunktionen der sieben verwendeten Features und einem offset. Hier sei nochmal darauf hingewiesen, dass Features wie *hasClicked* oder *clickCount* nicht verwendet werden können, da die Views aufgrund der Problematik der Datenerhebung nicht berücksichtigt werden. Die verwendeten Einflussgrößen sind also der Wochentag und die Stunde des jeweiligen Kontaktpunktes, die Art des aktuellen und der letzten zwei Kontakte, sowie die Dauer seit dem vorherigen Kontakt und die Gesamtdauer seit dem ersten bis zum jetzigen Kontakt. Die Funktionen $f_{.,p}$ können theoretisch für verschiedene Positionen komplett unterschiedliche Formen annehmen. Es sei ausdrücklich darauf hingewiesen, dass $f_{campaignLast,p}(campaign_{i,p-1})$ und $f_{campaign,p-1}(campaign_{i,p-1})$ zwei unterschiedliche Funktionen sind. Erstere gibt den Einfluss der Art des vorherigen Kontaktpunktes auf die Konvertierungswahrscheinlichkeit an der Position p wieder und zweitere den Einfluss der Art des aktuellen Kontaktpunktes auf die Konvertierungswahrscheinlichkeit an der Position $p-1$. Unter der Annahme, dass der Einfluss der Features an den verschiedenen Positionen ähnlich ist, fließt zusätzlich noch die Vorhersage des Modells der vorherigen Position als offset mit ein. Dadurch konnten die Ergebnisse besonders an späteren Positionen, an denen weniger Daten vorhanden sind, deutlich verbessert werden. Ergebnisse ohne offset sind im elektronischen Anhang zu finden.
Der offset fällt für Position eins weg, da es noch keine Vorhersagen eines vorherigen Modells gibt. Außerdem sind für Position eins die Features *campaignLast*, *campaignLast2*,

timeSinceLast und *timeSinceFirst* offentsichtlich noch nicht vorhanden, so dass diese ebenfalls wegfallen. An Position zwei ist *campaignLast2* noch nicht vorhanden und *timeSinceLast* und *timeSinceFirst* sind hier identisch, so dass nur eine der beiden berücksichtigt wird. Ab Position Drei ergibt sich der Prädiktor dann exakt so wie in (9) dargestellt. Algorithmus 1 enthält Pseudo-Code, der das Vorgehen beim Gradient Boosting erläutern

Algorithmus 1 Gradient Boosting

Setze Startwert für $f_{0p}(x_{ip})$
for $m = 1 : n.trees$ **do**
  Setzte $\lambda_{ip}(x_{ip}) = \frac{\exp(f_{m-1,p}(x_{ip}))}{1+\exp(f_{m-1,p}(x_{ip}))}$
  for $i = 1 : N_p$ **do**
    $r_{imp} = -\frac{\partial L(y_{ip}, f_{m-1,p}(x_{ip}))}{\partial f_{m-1,p}(x_{ip})} = y_{ip} - \lambda_{ip}(x_{ip})$
  end for
  $\theta_{mp} = \arg\min_\theta \sum_{i=1}^{N_p} (r_{imp} - h(x_{ip}, \theta))^2$
  $\beta_{mp} = \arg\min_\beta \sum_{i=1}^{N_p} L(y_{ip}, f_{m-1,p}(x_{ip}) + \beta h(x_{ip}, \theta_{mp}))$
  $f_{mp}(x_{ip}) = f_{m-1,p}(x_{ip}) + \beta_{mp} h(x_{ip}, \theta_{mp})$
end for

soll. Dieser muss für jedes $p = 1, ..., 25$ durchgeführt werden. Zunächst muss ein Startwert des Prädiktors f_{0p} festgelegt werden. Daraufhin werden folgende Schritte für $m = 1$ bis *n.trees* iteriert. Für jede Beobachtung i werden die Pseudo-Residuen r_{imp} berechnet, die die Richtung des negativen Gradienten angeben. Die Pseudo-Residuen entsprechen also der Richtung des steilsten Abstiegs der Verlustfunktion. An die Pseudo-Residuen wird ein Basis-Lerner $h(x_{ip}, \theta_{mp})$, in diesem Fall ein Entscheidungsbaum, so angepasst, dass er den negativen Gradienten so gut wie möglich approximiert. Bildlich gesprochen wird das Modell also in die Richtung der größten Verringerung des Verlusts verschoben. Da als Basis-Lerner Stümpfe verwendet werden, wird die Verbesserung des Modells in einem Boosting-Schritt durch nur eines der sieben Features erklärt. Per Line-Search wird daraufhin die optimale Schrittweite β_{mp} berechnet und das Modell wird geupdatet mit $f_{mp}(x_{ip}) = f_{m-1,p}(x_{ip}) + \beta_{mp} h(x_{ip}, \theta_{mp})$. Nach *n.trees* Iterationen endet der Algorithmus. Durch die Verringerung des Verlusts in jeder Iteration kann es vor allem bei einer hohen Anzahl von Iterationen zu Overfitting kommen. Deshalb wird mittels Kreuzvalidierung die optimale Anzahl an Iterationen ausgewählt. Das heißt $\hat{f}(x_{ip}) = f_{m\_opt,p}(x_{ip})$ wird als Ergebnis verwendet. Das Ensemble der Splits bildet für jedes der Feautures eine Treppenfunktion.

Parameter des Modells

Das Modell wurde mit dem Paket *gbm* [16] in *R* [18] berechnet und mit Hilfe der R-Pakete *foreach* [3] und *doSNOW* [2] parallelisiert. Die Modelle für die einzelnen Positionen können allerdings nur dann parallelisiert werden, wenn kein offset benützt wird. Für die Anwendung des Modells müssen noch einige Parameter eingestellt werden.
Zunächst wurden die Daten in Trainings- und Testdaten aufgeteilt, sodass Trainings- und Testdaten jeweils die Hälfte der gesamten Daten ausmachen. Da die Anzahl der

nicht-konvertierten Funnels deutlich überwiegt und einige Kampagnen vergleichsweise selten in den Daten auftreten, wurden die Trainingsdaten stratifiziert bezüglich konvertierter beziehungsweise nicht-konvertierter Funnel und der Variable *Campaign* gezogen. Außerdem wurde die Länger der Funnels berücksichtigt, so dass Trainings- und Testdaten positionsübergreifen klar getrennt sind und das Verhältnis von eins zu eins trotzdem eingehalten wird. Das heißt, dass ein Beobachtung, die an Position 1 in den Trainingsdaten enthalten ist auch an allen späteren Positionen in den Trainingsdaten ist. Das selbe gilt für die Testdaten. Dies ist wichtig, damit Trainings- und Testdaten bei der Anwendung des offsets nicht vermischt werden. Anhand der Trainingsdaten wurde das Modell gefittet. Die Testdaten dienen der späteren Bewertung der Prognosegüte des Modells. Die maximale Anzahl der Bäume *n.trees* wurde gleich 3000 gesetzt, wobei für die Präsentation der Ergebnisse die optimale Anzahl an Bäumen mittels 5-facher Kreuzvalidierung gewählt wurde.
Zusätzlich zu der Beschränkung der Iterationen, wird dem Overfitting auch durch einen Shrinkage-Parameter μ entgegen gewirkt. Dieser bewirkt, dass nicht die optimale Schrittweite β_{mp} in jeder Iteration gegangen wird, sondern nur ein Bruchteil dieser Schrittweite. Es wird also durch $f_{mp}(x_{ip}) = f_{m-1,p}(x_{ip}) + \mu\beta_{mp}h(x_{ip}, \theta_{mp})$ geupdatet, wobei μ gleich 0.01 gewählt wurde, was einem üblichen Wert für diesen Parameter entspricht. Bei der Wahl von μ und *n.trees* muss stets die Rechenzeit im Auge behalten werden.
Wie bereits erwähnt, wurden als Basis-Lerner Stümpfe gewählt. Das wurde durch die Festlegung von *interaction.depth* auf 1 realisiert. Mit einer höheren *interaction.depth* ließen sich auch Interaktionen modellieren. Die Stärke von solchen Interaktionen lassen sich beispielsweise mit Friedmans h-Statistik [10] untersuchen. Die Einführung von Interaktionen führt bei den vorliegenden Daten aber zu einer drastischen Verschlechterung der Prognosegüte und schwer interpretierbaren Ergebnissen. Bei Interesse sind Ergebnisse diesbezüglich im elektronischen Anhang zu finden.
Der bis hierhin beschriebene Algorithmus entspricht lediglich dem Gradient Boosting. Wenn man in jeder Iteration allerdings nur einen Teil der Daten verwendet, spricht man vom Stochastic Gradient Boosting. Diese Anpassung führt meist zur einer Verbesserung der Ergebnisse [8], wobei noch nicht geklärt ist, wie genau es zu dieser Verbesserung kommt. In diesem Modell wurde die *bag.fraction*, das heißt der Anteil der verwendeten Daten in jedem Schritt, auf 0.5 gesetzt.

Output

Als Ergebnis ist für jede Beobachtung i und jede Position p ein Wert $\hat{f}(x_{ip})$ gegeben. Daraus können die Hazardraten durch Rücktransformation (10) berechnet werden.

$$\hat{\lambda}_{ip} = \frac{\exp(\hat{f}(x_{ip}))}{1 + \exp(\hat{f}(x_{ip}))} \tag{10}$$

Diese geben die Wahrscheinlichkeit für Beobachtung i an Position p zu konvertieren an. Da die nicht-konvertierten Funnels in den Daten deutlich überwiegen, sind diese Wahrscheinlichkeiten sowohl für konvertierte als auch für nicht-konvertierte Funnels sehr niedrig. Deshalb kommt der Bayes-Klassifikator zur Beurteilung der Prognosegüte nicht in

Frage, da dieser schlicht alle Beobachtungen als nicht konvertiert vorhersagen würde. Es muss also ein geeigneter Schwellenwert gefunden werden, um die Prognosegüte des Modells zu bewerten. Wenn die Hazardrate einer Beobachtung größer ist als der Schwellenwert, so wird diese als konvertiert vorhergesagt und sonst als nicht-konvertiert. Dafür werden die Kenngrößen Sensitivität und Spezifität benötigt. Die Sensitivität ist die Richtig Positiv Rate, das heißt der Anteil der wirklich Konvertierten unter allen, die als konvertiert vorhergesagt werden, gegeben ein spezifischer Schwellenwert. Die Spezifität ist die Richtig Negativ Rate, das heißt der Anteil der Nicht-Konvertierten unter allen, die als nicht-konvertiert vorhergesagt werden. Diese Kennwerte wurden für ein Gitter von potentiellen Schwellenwerten berechnet.
Die ROC (Receiver Operating Characteristics)-Kurve erlaubt die simultane Betrachtung dieser Schwellenwerte. Diese bildet die Falsch Positiv Rate, das heißt 1 minus Spezifität, auf der x-Achse und die Sensitivität auf der y-Achse ab. Die ROC-Kurve hat ihren Ursprung im Punkt $(0, 0)$ und steigt von dort monoton an bis $(1, 1)$. Wenn Falsch Positiv Rate und Richtig Positiv Rate für alle Schwellenwerte gleich ist, so kann das Modell nicht zwischen konvertiert und nicht-konvertiert unterscheiden und die ROC-Kurve ist die Diagonale zwischen $(0, 0)$ und $(1, 1)$. Ein perfektes Modell, dass alle Beobachtungen korrekt zuweist geht durch den Punkt $(0, 1)$. Je größer die Fläche zwischen der Diagonalen und der ROC-Kurve ist, desto besser ist somit auch das Modell.
An dieser Stelle kommt die AUC (Area Under the Curve) ins Spiel. Sie ist definiert als die Fläche unter der ROC-Kurve (11). Wenn das Modell nicht zwischen konvertiert und nicht-konvertiert unterscheiden kann, so ist die AUC gleich 0.5 und die ROC ist, wie bereits erwähnt die Diagonale. Ein AUC von 1 entspricht einem perfekten Modell. Die AUC wird interpretiert als die Wahrscheinlichkeit, dass bei einer konvertierten Beobachtung die Hazardrate größer ist als bei einer nicht-konvertierten Beobachtung.

$$AUC = \int_0^1 ROC(t)\,\mathrm{d}t. \tag{11}$$

Ein weiterer Kennwert, der die Prognosefunktion beschreibt, ist die Relative Wichtigkeit der Einflussgrößen [9]. Sie gibt die Stärke des Einflusses der Features auf die Prognoseregel an. $\hat{I}^2_{jp}(m)$ (12) ist die Verbesserung, die an Position p in Iteration m durch die Variable j realisiert wird, wobei $\hat{i}_{mp}1_{jmp}$ eben diese Verbesserung ist und 1_{jmp} die Indikatorfunktion, die angibt, ob in Iteration m das Feature j als Splitvariable genutzt wurde. $\hat{I}^2_{jp}$ (13) mittelt für jedes Feature j über alle Iterationen beziehungsweise Bäume und die Relative Wichtigkeit $\hat{I}_{jp}$ (14) von Feature j ergibt sich dann aus der Wurzel dieses Wertes. Die Wichtigkeitswerte werden so skaliert, dass sie in der Summe den Wert 100 ergeben.

$$\hat{I}^2_{jp}(m) = \hat{i}_{mp}1_{jmp} \tag{12}$$

$$\hat{I}^2_{jp} = \frac{1}{M}\sum_{m=1}^{n.trees} \hat{I}^2_{jp}(m) \tag{13}$$

$$\hat{I}_{jp} = \sqrt{\hat{I}^2_{jp}} \tag{14}$$

Neben der Information zur Stärke des Einflusses eines Features, interessiert natürlich auch die Art dieses Einflusses, das heißt der marginale Effekt eines Features [9]. Das Ziel ist somit ein Plot, der den Zusammenhang zwischen dem Feature j und der Prognosefunktion $\hat{f}$ an Position p darstellt. Da die Prognosefunktion von mehreren Features abhängt, müssen die marginalen Effekte berechnet werden. Das Feature x_{jp} ist eine Teilmenge aller Features $\{x_{1p}, ..., x_{np}\}$ und $x_{\setminus j,p}$ sei dessen Komplement. Damit kann die Prognosefunktion als $\hat{f}(x_p) = \hat{f}(x_{jp}, x_{\setminus j,p})$ geschrieben werden und wenn man auf bestimmte Werte von $x_{\setminus j,p}$ bedingt, kann die Prognosefunktion als Funktion von lediglich x_{jp} betrachtet werden.

$$\hat{f}_{x_{\setminus j,p}}(x_{jp}) = \hat{f}(x_{jp}|x_{\setminus j,p}) \tag{15}$$

Die Form von $\hat{f}_{x_{\setminus j,p}}(x_{jp})$ hängt von dem für $x_{\setminus j,p}$ gewählten Wert ab. Wenn diese Abhängigkeit nicht zu stark ist, dann ist (16) eine gute Zusammenfassung des marginalen Effektes von x_{jp} auf die Prognosefunktion. Dabei ist $p_{\setminus j,p}(x_{\setminus j,p})$ die marginale Dichte von $x_{\setminus j,p}$ und $p(x_p)$ ist die gemeinsame Dichte der Features.

$$\bar{f}_{jp}(x_{jp}) = E_{x_{\setminus j,p}}(\hat{f}(x_p)) = \int \hat{f}(x_{jp}, x_{\setminus j,p}) p_{\setminus j,p}(x_{\setminus j,p}) \, \mathrm{d}x_{\setminus j,p}. \tag{16}$$

$$p_{\setminus j,p}(x_{\setminus j,p}) = \int p(x_p) \, \mathrm{d}x_{\setminus j,p}. \tag{17}$$

Die marginale Dichte kann aus den Trainingsdaten geschätzt werden, so dass (16) zu (18) wird.

$$\bar{f}_{jp}(x_{jp}) = \frac{1}{N} \sum_{i=1}^{N_p} \hat{f}(x_{jp}, x_{i,\setminus j,p}) \tag{18}$$

Die marginalen Effekte werden wiederum zu den Hazardraten rücktransformiert.

5 Sequential Pattern Mining

5.1 Überblick

Sequential Pattern Mining entdeckt häufige Subsequenzen, das heißt Teilfolgen, in Datenbanken. Sogenannte Sequenz-Datenbanken bestehen aus Transaktionen, die jeweils eines oder mehrere Items enthalten, welche der Zeit nach geordnet sind. Bei den gegebenen Daten ist eine Transaktion ein Kontaktpunkt, wobei das Item die Art des Kontaktpunktes ist. Ein Funnel setzt sich aus mehreren Transaktionen zusammen.
Ein Anwendungsfeld ist die Warenkorbanalyse. Angenommen es wird das Kaufverhalten in einem Supermarkt einen Monat lang beobachtet, dann könnte [Kunde 1, <(Brot, Milch), (Brot, Milch, Tee), (Zucker), (Milch, Salz)>]; [Kunde 2, <(Brot), (Milch, Tee)>] eine Beispiel-Datenbank sein. Kunde 1 war vier mal im beobachteten Monat im Supermarkt einkaufen, wobei Kunde 2 nur zweimal einkaufen war. Der Kunde kann nur eins

oder auch mehrere Items pro Besuch einkaufen. Im Falle von mehreren Items spricht man von Itemsets.
Web Usage Mining ist das am weitesten verbreitete Anwendungsfeld von Sequential Pattern Mining in der Literatur ([13, 21, 11]). Unter der Annahme, dass ein Internetnutzer nur eine Webseite an einem Zeitpunkt aufrufen kann, bestehen die Sequenzen nur aus einzelnen Items und enthalten keine Itemsets. Ist also eine Menge von Items $I = \{a, b, c, d, e\}$ gegeben, die beispielsweise verschiedene Webseiten repräsentieren, so könnte eine Datenbank mit zwei Nutzern folgendermaßen aussehen: [Kunde 1, <abedcab>]; [Kunde 2, <edcaa>] ([14]).
In den letzten zwei Jahrzehnten wurden im Forschungsfeld des Sequential Pattern Mining eine Vielzahl von Algorithmen entwickelt. Beispiele sind der HVSM- [19], der PrefixSpan- [17], der FS-Miner [5] und der PLWAP-Algorithmus [6]. Folglich stellt sich die Frage, welcher Algorithmus aus dieser großen Auswahl hier verwendet werden soll. Es gibt einige Arbeiten (beispielsweise [20, 7, 15, 12]), die Sequential Pattern Mining-Algorithmen, die auf Online-Tracking-Daten anwendbar sind, miteinander vergleichen. Diese konzentrieren sich allerdings nur auf die Techniken und Theorien, auf denen die jeweiligen Algorithmen basieren und bewerten sie anhand ihrer Leistung bei unterschiedlichen Datenstrukturen. Auf die Bewertung der eigentlichen Ergebnisse der Algorithmen wird dabei nicht eingegangen. Deshalb kann an dieser Stelle reinen Gewissens der SPADE-Algorithmus [22] verwendet werden, der der einzige Sequential Pattern Mining-Algorithmus ist, der in R implementiert ist. Außerdem schneidet er im Vergleich zu anderen Algorithmen stets solide ab.

5.2 Problemstellung

Das Problem wurde erstmals im Jahre 1995 in der Arbeit *Mining Sequential Patterns* [1] von Agrawal & Srikant vorgestellt. Es ist eine Datenbank D von Transaktionen gegeben, wobei jede Transaktion eine ID, den Zeitpunkt der Transaktion und die dazugehörigen Items enthält. Für jede ID können mehrere Transaktionen existieren allerdings nie zwei oder mehr Transaktionen mit dem selben Zeitpunkt.
Ein Itemset $i = (i_1 i_2 ... i_m)$ ist eine nichtleere Menge von Items i_j und eine Sequenz $s = s_1 \rightarrow s_2 \rightarrow ... \rightarrow s_n$ eine geordnete Liste von Itemsets s_j. Ohne Beschränkung der Allgemeinheit wird angenommen, dass die Menge der Items auf eine Menge von fortlaufenden ganzen Zahlen abgebildet wird.
Eine Sequenz $a_1 \rightarrow a_2 \rightarrow ... \rightarrow a_n$ heißt Subsequenz einer anderen Sequenz $b_1 \rightarrow b_2 \rightarrow ... \rightarrow b_m$ wenn ganze Zahlen $i_1 < i_2 < ... < i_n$ existieren, so dass $a_1 \subseteq b_{i_1}$, $a_2 \subseteq b_{i_2}$, ..., $a_n \subseteq b_{i_n}$ gilt. Beispielsweise ist $< (1)(58)(27) >$ Subsequenz von $< (15)(248)(358)(27) >$, da $(1) \subseteq (15)$, $(58) \subseteq (358)$ und $(27) \subseteq (27)$ aber $< (27) >$ keine Subsequenz von $< (2)(7) >$ und umgekehrt. Eine Sequenz heißt maximal in einer Menge von Sequenzen, wenn sie von keiner Sequenz in dieser Menge Subsequenz ist.
Die Transaktionen einer ID können als Sequenz verstanden werden. Jede Transaktion entspricht dabei einem Itemset und die nach den Zeitpunkten $T_1, T_2, ..., T_n$ geordneten Transaktionen entsprechen einer Sequenz $itemset(T_1) \rightarrow itemset(T_2) \rightarrow ... \rightarrow itemset(T_n)$. Der Support einer Sequenz s ist der Anteil der IDs, die s unterstützen,

das heißt deren Sequenz Subsequenz von s sind. Eine Sequenz mit k Items wird auch k-Sequenz genannt.
Das Problem des sequential pattern mining besteht darin, in einer Datenbank D unter allen existierenden Sequenzen die maximalen Sequenzen zu finden, deren support größer oder gleich einem festgelegten minimalen Support ist.

5.3 SPADE

Im Folgenden wird der SPADE Algorithmus [22] vorgestellt, der in dem R-Paket *arulesSequences* [4] implementiert ist. Dieser sucht häufige Sequenzen von Itemsets, das heißt er kann auch mit Daten umgehen, die mehrere Items pro Zeitpunkt enthalten. Da ein Benutzer nur eine Webseite zum exakt selben Zeitpunkt besuchen kann, ist diese Eigenschaft in diesem Fall nicht relevant. Die Daten werden vertikal bezüglich der IDs angeordnet. Die zweite Spalte enthält die Zeit des Kontaktes, die dritte Spalte die Anzahl der Items zu diesem Zeitpunkt und die vierte Spalte die eigentlichen Items. Bei den gegebenen Daten sind die Zeitpunkte die Positionen und damit pro ID von eins bis zur Anzahl der Positionen für die ID durchnummeriert. Die Anzahl der items in der dritten Spalte ist immer gleich eins und die vierte Spalte enthält pro Zeile nur ein Item (Tabelle 3).

ID	Position	Anzahl Items	Items
1	1	1	A
1	2	1	D
1	3	1	A
1	4	1	B
2	1	1	C
2	2	1	B
3	1	1	D
4	1	1	A
4	2	1	D
4	3	1	B

Tabelle 3: Beispiel für eine Input-Datenbank für den SPADE-Algorithmus

Berechnung des Supports

In diesem Abschnitt wird beschrieben, wie der Support einer Sequenz berechnet wird. Tabelle 4 enthält die ID-Listen der Items aus den Daten aus Tabelle 3. Item A beispielsweise, tritt für ID 1 an Position 1 und 3 und für ID 4 an Position 1 auf.
Jede Sequenz ist eine zeitabhängige Verknüpfung von Items und der Support der Sequenz kann mittels einer zeitabhängigen Verknüpfung der ID-Listen jedes Items in der Sequenz berechnet werden. In Tabelle 5 ist diese Vorgehen beispielhaft für die Sequenz

A		B		C		D	
ID	Position	ID	Position	ID	Position	ID	Position
1	1	1	4	2	1	1	2
1	3	2	2			3	1
4	1	4	3			4	2

Tabelle 4: ID-Listen für die items

$A \to D \to B$ dargestellt. Den Support erhält man dann, indem die Anzahl der einzigartigen IDs, die die Sequenz enthalten durch die Anzahl aller IDs in den Daten teilt.
Wenn man für jedes Item einer Sequenz die Zeitpunkte zusammenfasst, wie in Tabelle

A		AD			ADB			
ID	Pos(A)	ID	Pos(A)	Pos(D)	ID	Pos(A)	Pos(D)	Pos(B)
1	1	1	1	2	1	1	2	4
1	3	4	1	2	4	1	2	3
4	1							

Tabelle 5: Temporale Verknüpfung

5, dann wird zu viel Speicherplatz verbraucht. Um diesem Problem entgegen zu wirken, wird das Korollar ausgenutzt, das besagt, dass eine Sequenz der Länge k immer aus der Kombination ihrer lexikographisch ersten zwei Subsequenzen der Länge $(k-1)$ gebildet werden kann. Damit werden für jede Sequenz die ID-Spalte und nur eine Spalte für die Zeitpunkte benötigt.
Dies führt zu einer Speicherreduktion da die zwei ersten Sequenzen der Länge $(k-1)$, X_1 und X_2, einer Sequenz X einen Prefix der Länge $(k-2)$ teilen. Damit sind auch die Zeitpunkte im Prefix gleich und X_1 und X_2 unterscheiden sich lediglich bezüglich des letzten items.
In dem Beispiel mit $X = (A \to D \to B)$ entsteht X durch den temporalen join von $X_1 = (A \to D)$ und $X_2 = (A \to B)$. X_1 entsteht durch das Wegfallen des letzten Items von X und X_2 durch das Wegfallen des vorletzten Items. Im Falle längerer Sequenzen wird dieses Vorgehen rekursiv durchgeführt bis die einzelnen Items übrig bleiben. Sei X eine Subsequenz von Y, so gilt, dass die Mächtigkeit der ID-Liste von Y kleiner oder gleich der Mächtigkeit der ID-Liste von X sein muss. Das führt zu schnellen Joins und schnellen Berechnungen des Supports.

Prefix-basierte Klassen

Sei $p : (S, N) \to S$ eine Funktion, wobei S die Menge der Sequenzen und N eine Menge von nichtnegativen natürlichen Zahlen sind. $p(X, k) = X[1 : k]$ gibt den Prefix von X der Länge k zurück. Die Äquivalenz Relation θ_k wird wie folgt definiert. Für alle $X, Y \in S$ ist X mit Y verwandt unter θ_k, in Zeichen $X \equiv_{\theta_k} Y$, genau dann wenn $p(X, k) = p(Y, k)$.

Das heißt, zwei Sequenzen sind in der selben Klasse bezüglich θ_k, wenn sie den selben Prefix der Länge k teilen.

Suchstrategien

Breadth-First und Depth-First sind effiziente Suchstrategien, um häufige Sequenzen innerhalb von Eltern-Klassen zu finden. Beide basieren auf der rekursiven Zerlegung der Eltern-Klassen in kleinere Klassen die anhand der Äquivalenz Relation θ_k gebildet werden und dem resultierenden Verbund von Äquivalenzklassen.
Bei der Breadth-First-Suche (BFS) wird der Verbund der Äquivalenklassen, die durch die rekursive Anwendung von θ_k entstehen, bottom-up untersucht. Das heißt alle Klasse einer Ebene werden verarbeitet, bevor man zur nächsten Ebene vorgeht. Der Vorteil dieser Vorgehensweise ist, dass man beispielsweise die Menge der 2-Sequenzen kennt bevor die 3-Sequenzen gebildet werden. DFS verbraucht weniger Arbeitsspeicher als BFS.
Bei der Depth-First-Suche (DFS) werden alle Äquivalenzklassen eines Pfades bearbeitet, bevor man zum nächsten Pfad übergeht. In R sind beide Suchstrategien implementiert, wobei in dieser Arbeit DFS verwendet wurde.

Verknüpfung von ID-Listen

Hier soll die Verknüpfung von zwei ID-Listen etwas genauer erklärt werden. Sei $P = [A \rightarrow B]$ eine Äquivalenzklasse mit den Elementen $\{P \rightarrow B, P \rightarrow C, P \rightarrow E\}$. Da bei Online-Tracking-Daten keine Itemsets möglich sind, reduziert sich die Anzahl der mögliche Fälle bei der Verknüpfung. Werden $P \rightarrow B$ und $P \rightarrow C$ verknüpft, so gibt es nur zwei mögliche Ergebnisse, $P \rightarrow B \rightarrow C$ oder $P \rightarrow C \rightarrow B$. Ein Spezialfall ist die Verknüpfung von $P \rightarrow B$ mit sich selber. Dabei kann nur die Sequenz $P \rightarrow B \rightarrow B$ entstehen.
Tabelle 6 enthält ein Beispiel zur Verknüpfung von den ID-Listen von $P \rightarrow B$ und $P \rightarrow C$. Um die ID-Liste von $P \rightarrow B \rightarrow C$ zu ermitteln muss überprüft werden, ob es zeitliche Beziehungen gibt. Das heißt, für jedes $(ID_i, Position_i)$-Paar aus der ID-Liste von $P \rightarrow B$ wird überprüft, ob es in der ID-Liste von $P \rightarrow C$ ein $(ID_j, Position_j)$-Paar gibt mit $Position_j > Position_i$ und $ID_i = ID_j$. Wenn dies der Fall ist, so enthält ID_i die Sequenz $P \rightarrow B \rightarrow C$ und das Paar $(ID_i, Position_k)$ wird dessen ID-Liste hinzugefügt. Die ID-Liste von $P \rightarrow C \rightarrow B$ wird analog ermittelt.
Da nur Sequenzen innerhalb einer Klasse, die den selben Prefix besitzen, verknüpft werden, muss nur die Position des letzten Items berücksichtigt werden.

Der Algorithmus

Algorithmus 2 enthält die oberflächliche Struktur von SPADE. Die Hauptschritte sind die Berechnung der häufigen 1-Sequenzen und 2-Sequenzen, die Zerlegung in Prefix-basierte Eltern-Äquivalenzklassen und die Berechnung aller anderen häufigen Sequenzen mittels BFS oder DFS. Im Folgenden sollen die einzelnen Schritte etwas genauer erläutert werden.
Aus der vertikalen Datenbank (Tabelle 3) erhält man die ID-Listen (Tabelle 4). Der Sup-

$P \to B$		$P \to C$		$P \to B \to C$		$P \to C \to B$	
ID	Position	ID	Position	ID	Position	ID	Position
1	2	1	7	1	7	8	5
1	3	1	8	1	8	8	8
1	4	3	2	8	3	13	5
4	6	5	7	8	4	13	7
7	4	8	3	8	5		
8	2	8	4	8	8		
8	3	8	5				
8	5	8	8				
8	8	11	3				
13	5	13	2				
13	7	16	8				
15	6	20	2				
17	2						
20	2						

Tabelle 6: Verknüpfung von ID-Listen

Algorithmus 2 SPADE(min_sup, D)

$F_1 = \{$*frequent items or 1-sequences*$\}$
$F_2 = \{$*frequent 2-sequences*$\}$
$\epsilon = \{$*equivalence classes* $[X]_{\theta_1}\}$
for all [X] $\in \epsilon$ **do**
  Enumerate-Frequent-Seq([X])
end for

port der 1-Sequenzen entspricht dann jeweils der Anzahl der unterschiedlichen IDs in den ID-Listen. Wenn der Support für ein Item größer als der minimal geforderte Support ist, so wird dieses Item als häufige 1-Sequenz abgespeichert. Dieser Vorgang benötigt nur einen Scan der Datenbank.
Sei $N = |F_1|$ die Anzahl der häufigen Items und A die mittlere ID-Listen Größe in Bytes. Ein naiver Ansatz ist, alle $\binom{N}{2}$ ID-List-Verknüpfungen durchzuführen und dann jeweils den Support zu berechnen. Dafür müssten $A * N * (N-1)/2$ Bytes gelesen werden, was ungefähr $N/2$ Datenscans entspricht.
Eine effizientere Alternative ist eine nahtlose Transformation der vertikalen ID-Listen in eine horizontale Datenbank. Tabelle 7 stellt diese Transformation für das Beispiel aus Tabelle 4 dar. Die (ID, Position)-Paare kategorisiert nach Items werden zu (Item, Position)-Paaren kategorisiert nach ID transformiert. Anhand der horizontalen Datenbank kann F_2 einfach berechnet werden. Es wird eine Liste aller 2-Sequenzen in jeder ID-Liste gebildet und in einem zweidimensionalen Array werden die Anzahlen der 2-Sequenzen geupdated.
Daraufhin wird die Menge ϵ aller Äquivalenzklassen $[X]_{\theta_1}$ gebildet. Das heißt alle 2-

ID	(Item, Position)-Paare
1	(A,1)(A,3)(B,4)(D,2)
2	(B,2)(C,2)
3	(D,1)
4	(A,1)(B,3)(D,2)

Tabelle 7: Transformation in horizontale Datenbank

Sequenzen, die das selbe erste Item haben, werden zu einer Klasse zusammengefasst.
Im letzten Schritt werden dann für alle Klassen $[X]$ aus ϵ die häufige Sequenzen ermittelt. Dies geschieht durch die Verknüpfung der ID-Listen von allen Elementen aus $[X]$ und der Überprüfung, ob diese den minimalen Support erfüllen. Die als häufig ermittelten Sequenzen liefern die ID-Listen zur Verknüpfung auf der nächsten Ebene. Dieser Prozess wird fortgesetzt bis alle häufigen Sequenzen ermittelt wurden. Dieser Suchvorgang kann mittels BFS oder DFS bewerkstelligt werden.

Pruning

Der SPADE-Algorithmus unterstützt Pruning von Sequenzen. Sei α_1 das erste Element einer Sequenz α. Vor der Verknüpfung einer neuen k-Sequenz β, wird überprüft, ob alle k Subsequenzen der Länge $(k-1)$ von β häufig sind. Ist dies nicht der Fall, kann β keine häufige Sequenz sein und wird nicht weiter berücksichtigt. Ein Problem dabei ist, dass nur $(k-1)$ der k Subsequenzen in der selben Klasse wie β sind.
Man betrachte das folgende Beispiel mit der 4-Sequenz $\beta = (A \rightarrow B \rightarrow C \rightarrow D)$. Die 4 Subsequenzen der Länge 3 sind $(A \rightarrow B \rightarrow C)$, $(A \rightarrow B \rightarrow D)$, $(A \rightarrow C \rightarrow D)$ und $(B \rightarrow C \rightarrow D)$, wobei letztere nicht der selben Klasse angehört wie β. Die ersten drei Subsequenzen liegen in der Klasse $[A]$ und die letzte in der Klasse $[B]$. Wenn die Klasse $[B]$ bereits bearbeitet wurde, liegen alle Informationen zum Pruning bereits vor. Ansonsten kann kein Pruning durchgeführt werden, wobei partielles Pruning, dass nur auf den Subsequenzen aus Klasse $[A]$ basiert möglich ist.
Ein Nachteil des Pruning ist, dass alle häufigen Sequenzen der vorherigen Ebenen in einer Art Hashtabelle gespeichert werden müssen. Obwohl diese Art von Pruning eine große Anzahl von potentiellen Sequenzen ausschließen kann, haben Simulationsstudien [22] gezeigt, dass dadurch die Laufzeit nicht verbessert wird. Das liegt vor allem daran, dass die Verknüpfung von ID-Listen besonders für längere Sequenzen sehr schnell ist. Das heißt, dass die Verknüpfung ungefähr gleich schnell ist wie das Überprüfen, ob alle Subsequenzen häufig sind. Zudem kann bei einer großen Anzahl von häufigen Sequenzen der virtuelle Arbeitsspeicher beim Laden eben dieser überschritten werden. Deshalb ist das Pruning von Sequenzen in R nicht implementiert.

6 Visualisierung SPM

6.1 Idee und Erklärung

– bessere Einleitung/Überleitung
SPM versucht die Muster in den Abfolgen von Kampagnen zu erkennen.

Wenn man die Kampagnen für jeden Funnel und allen touchpoint aneinanderreiht bekommt man viele Ketten bzw. Abfolgen an Kampagnen. Viele Funnels haben die gleiche Kampagne an der gleichen Position und haben sogar identische Abschnitte. Betrachtet man nun die Kampagnen an einer bestimmten Position als Knotenpunkt und die Reihenfolge als Verbindungen zwischen diesen Knotenpunkt, ergibt sich ein Netzwerk mit Nodes und Edges. Mit dem rgexf Paket in R können wir dieses Netzwerk als .gexf format beschreiben, welches im Prinzip ein XML Format ist. Durch das Tool Gephi, lesen wir diese Datei ein un erzeugen so einen Graphen der unsere Nodes und Edges darstellt. Dabei werden auch einige Parameter übergeben, wie z.B. ein relative Edge weight, dass uns aus welchen anteilen sich eine Node zusammensetzt. Die nullte Ebene beschreibt den Startpunkt, die erste Ebene sind dann alle Werbeformen der ersten Position. Zusätlich wurden noch ab der ersten Ebene zwei neue Nodes, Success und Fails, erzeugt die die Konvertierung bzw nicht Konvertierung darstellt. Da die räumliche Anordnung nach dem ersten einlesen der .gexf Datei noch willkürlich ist, wenden wir einesn Algorithmus an der die Nodes räumlich anordnet mit Bezug auf den Parametern die wir übergeben haben. Dieser Algorithmus heißt Force Atlas 2 und berechnet die Position der Nodes aufgrund der Abstoßungskraft der Nodes und der Anziehungskraft der Edges. Das bedeutet die genau Position der Nodes sind immer noch ohne Aussagekraft, dafür aber die relative Anordnung untereinander. Das heißt stärker verbundene Nodes liegen näher beieinander als schwach verbundene. Die logische Konsequenz daraus ist, dass der Startpunkt näher an der ersten Ebene liegt, weil er ja nur an dieser verknüpft ist. Somit ergibt sich eine lineare Struktur, die auch Sinn macht, weil es keine Verbindungen zwischen Ebenen gibt die mehr als zwei Schritte voneinander entfernt sind.

6.2 Graphen

7 Ergebnisse

7.1 Zeitdiskretes Survival-Modell

Die in diesem Kapitel präsentierten Ergebnisse basieren jeweils auf der bestmöglichen Anzahl an Iterationen. Diese wurden mittels 5-facher Kreuzvalidierung ermittelt und sind in Abbildung 12 dargestellt. Auf der x-Achse ist die Position von 1 bis 25 aufgetragen und auf der y-Achse die Optimale Iterationsanzahl, die durch $n.trees = 3000$ nach oben begrenzt ist. Es ist zu erkennen, dass für die Positionen 1 bis 4 über 2500 Stümpfe für die Ergebnisse verwendet werden. Dann fällt die Kurve sehr schnell ab bis sie sich bei

ungefähr 500 Iterationen einpendelt. Dieser Abfall ist dadurch zu erklären, dass mit steigender Position die Datenmenge sinkt.

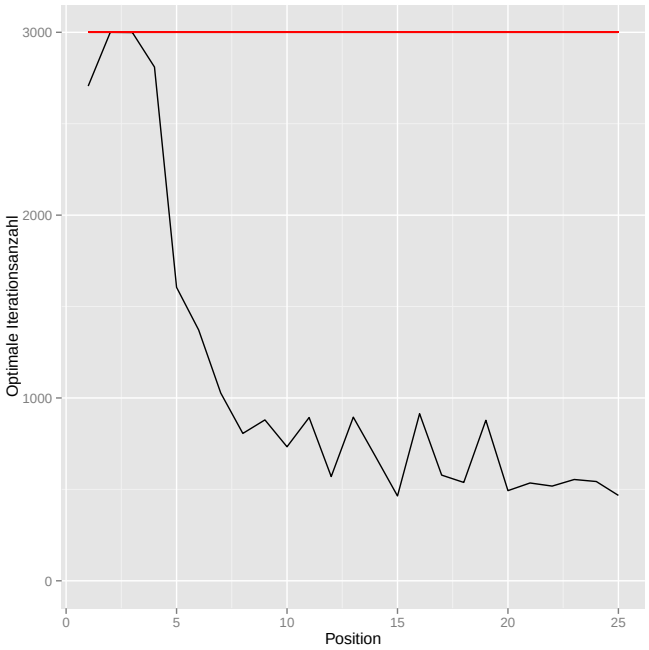

Abbildung 12: Optimale Iterationsanzahl des Stochastic Gradient Boosting

Der relative Einfluss der Features gibt dessen Wichtigkeit bei der Erstellung der Prädiktorfunktion an und ist in Abbildung 13 dargestellt. Auf der x-Achse ist erneut die Position aufgetragen und auf der y-Achse der relative Einfluss. Der relative Einfluss aller Features an einer Position ergibt summiert jeweils 100 und die Balken sind der Größe nach geordnet, das heißt die wichtigsten Variablen sind unten abgebildet. Die Farben der Balken werden in der Legende rechts neben der Abbildung erläutert.
An Position 1 sind nur drei Features vorhanden, wobei *Campaign*, das heißt die Art des Kontaktpunktes, für circa 90% der Minimierung der Verlustfunktion verantwortlich ist. Die Variablen *Hour* und *Weekday* haben kaum Einfluss. An Position 2 ist die Kampagne immer noch das stärkste Feature, wobei hier auch die Kampagne des ersten Kontaktpunktes und die Zeit, die seit dem ersten Kontaktpunkt vergangen ist, Einfluss haben.

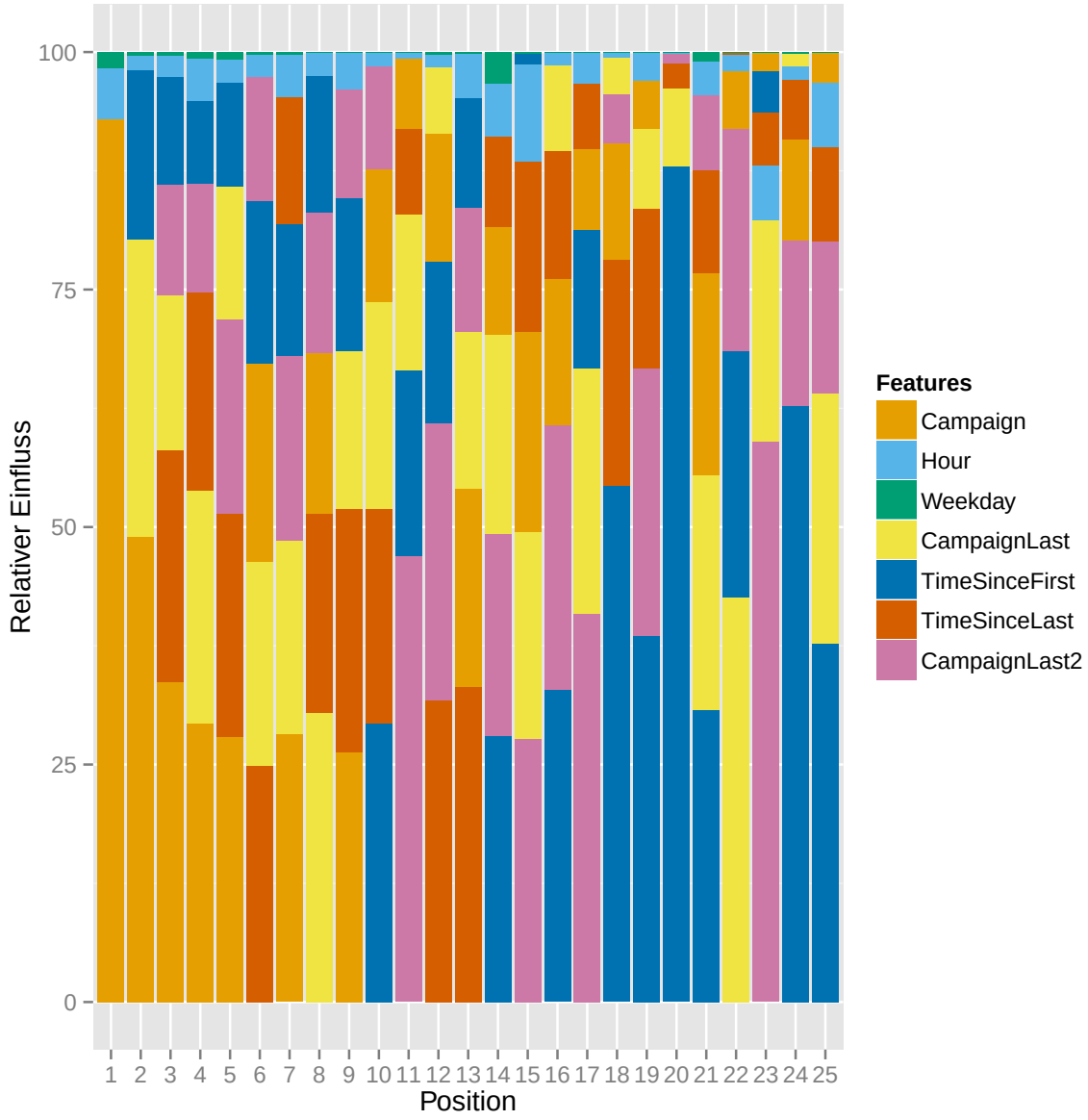

Abbildung 13: Wichtigkeit der Variablen

An den Positionen 8 und 22 hat die Kampagne des letzten Kontaktpunktes den stärksten Einfluss und damit selbstverständlich auch einen höheren Einfluss als die Kampagne des aktuellen Kontaktpunktes, die insbesondere an den höheren Positionen oft nur noch einen geringen Einfluss hat. Mit steigender Position wird auch die Kampagne des vorletzten Kontaktes wichtiger und ist öfters das wichtigste Feature. Die Zeit, die seit dem vorherigen Kontaktpunkt vergangen ist, ist bereits an Position 3 zweitwichtigstes Feature und spielt auch für die folgenden Positionen eine Rolle. Später nimmt die Wichtigkeit dieses Features allerdings ab und die Zeit, die seit dem ersten Kontaktpunktpunkt vergangen ist, wird bedeutend wichtiger. Die Features *Hour* und *Weekday* spielen po-

sitionsübergreifend kaum eine Rolle.
Nachdem die Wichtigkeit der Variablen für die Klassifizierung in konvertierte und nichtkonvertierte Funnel präsentiert wurde, ist nun die Art dieses Einflusses interessant. Dieser wird im folgenden für einige Positionen dargestellt, wobei die Variablen dafür nach positionsspezifischer Wichtigkeit ausgesucht werden und das Augenmerk vor allem auf die niedrigeren Positionen fällt, da dort mehr Daten vorhanden sind. Folglich werden die Features *Hour* und *Weekday* garnicht präsentiert, das sie kaum einen Einfluss haben. Die Kampagne des vorletzten Kontaktpunktes *CampaignLast*2 wird an dieser Stelle ebenfalls nicht berücksichtigt, da die Ergebnisse für *Campaign* beziehungsweise *CampaignLast* oft sehr ähnlich sind. Im elektronischen Anhang sind die marginalen Effekte aller Features für jeweils jede Position zu finden.
In Abbildung 14 sind die marginalen Effekte der Variable *Campaign* für die Positionen eins bis vier, von links oben nach rechts unten, aufgetragen. *Campaign* ist an den ersten vier Positionen das wichtigste Feature. Mit steigender Position verliert die Variable deutlich an Einfluss. Auf der x-Achse ist die Art des aktuellen Kontaktpunnktes aufgetragen und auf der y-Achse der Marginale Effekt.
An Position eins haben die Kampagnen *Affiliate - Rest*, *E-Mailing* und *SEM - Brand* im Vergleich zu den restlichen Kampagnen einen starken, positiven Einfluss auf die Konvertierungswahrscheinlichkeit.
Das heißt das Bereitstellen von Zinsvergleichen, welche das Zinsangebot der Interhyp AG mit deren Wettbewerbern im Vergleich darstellt, durch die Partner unter *Affiliate - Rest* scheint oft an Position 1 bereits zur Konvertierung zu führen. Die Partner unter *Affiliate - Partnerprogramm*, die hauptsächlich Rechner der Interhyp AG auf ihren Seiten einbinden, teilweise aber auch Banner schalten oder Verlinkungen in Texten unterbringen, haben an Position 1 einen deutlich geringeren Effekt. *E-Mailing* sind Mails, die an Interessenten, die schon einen Antrag gestellt haben oder ein Infopaket angefordert hatten, versendet werden. Da dieses Klientel sich offentsichtlich bereits intensiv mit einer Baufinanzierung beschäftigt hat, macht es Sinn, dass diese Mails schon nach wenigen Kontakten häufig zu einer Konvertierung führen und dieser Effekt für spätere Positionen schwächer ist. SEM sind bezahlte Suchergebnisse, hauptsächlich auf Google. *SEM - Brand* bedeutet, dass der Suchbegriff das Wort *Interhyp* enthielt. Dieser Effekt ist deutlich größer als für *SEM - Generisch* beziehungsweise *SEM - Remarketing*, wobei ersteres bedeutet, dass etwas wie *Baufinanzierung* gesucht wurde und zweiteres, dass der potentielle Kunde bereits zuvor auf der Seite von Interhyp war und deshalb nochmal eine Einblendung mit Werbung der Interhyp AG bekommen hat.
An Position 2 stechen neben *Affiliate - Rest*, *SEM - Brand* und *E-Mailing* auch *Direct* und *Generic* durch ihre Marginalen Effekte leicht hervor. *Direct* bedeutet, dass jemand im Browser direkt *www.interhyp.de* eingegeben hat und *Generic*, dass jemand über einen unbezahlten Link zur Interhyp kam. Für die Positionen 3 und 4 ist vor allem *Generic* wichtig. *Display*, *Social Media*, *SEO* sowie die Kooperationen spielen, neben den bereits erwähnten, keine große Rolle für die ersten vier Positionen.

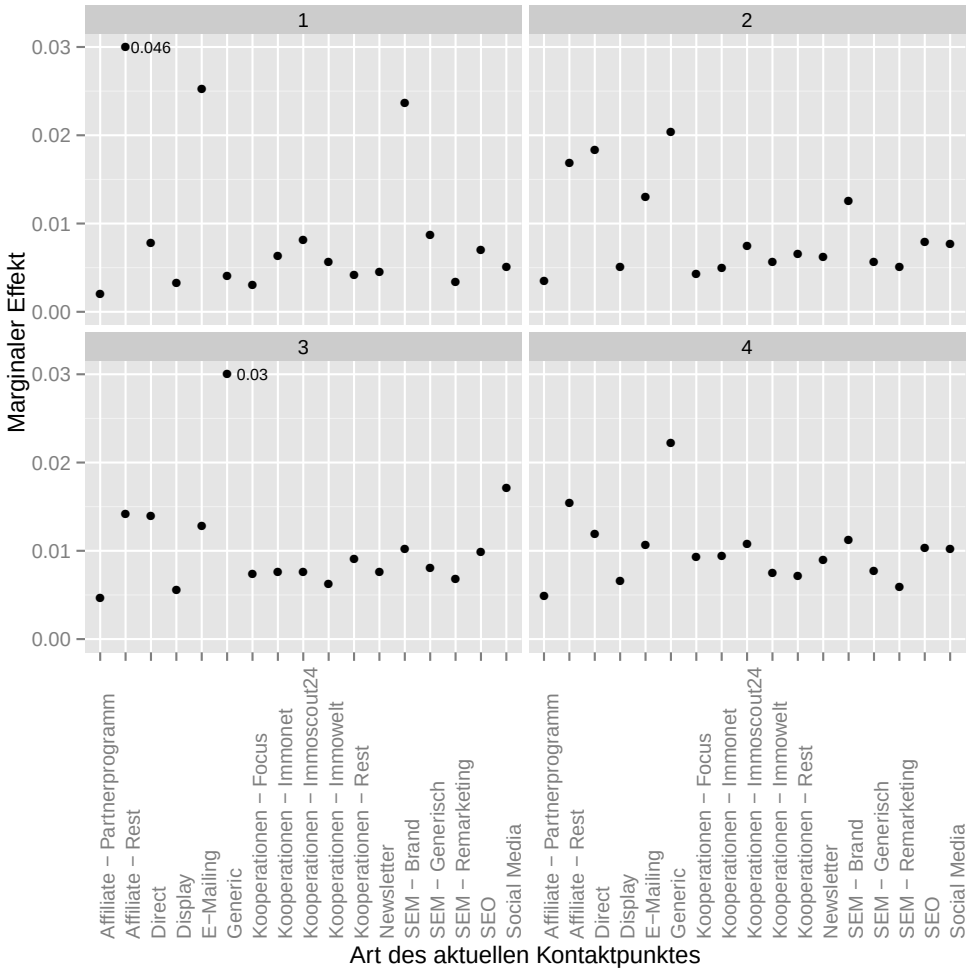

Abbildung 14: Marginaler Effekt des Features *Campaign* an den Positionen 1, 2, 3 und 4

An den Positionen 4, 6 und 7 und ist *CampaignLast* das zweitwichtigste Feature sowie an Position 8 das wichtigste. Dessen marginalen Effekte sind in Abbildung 15 dargestellt. Hier heben sich teilweise die selben Kategorien hervor, wobei die Unterschiede zwischen den Kampagnen kleiner sind. Überraschend ist, dass hier auch *Social Media* einen starken Effekt hat. Allerdings liegen hierfür kaum Daten vor, wie in Abbildung **AUF ABBILDUNG MIT CAMPAIGN VERWEISEN** zu erkennen ist.

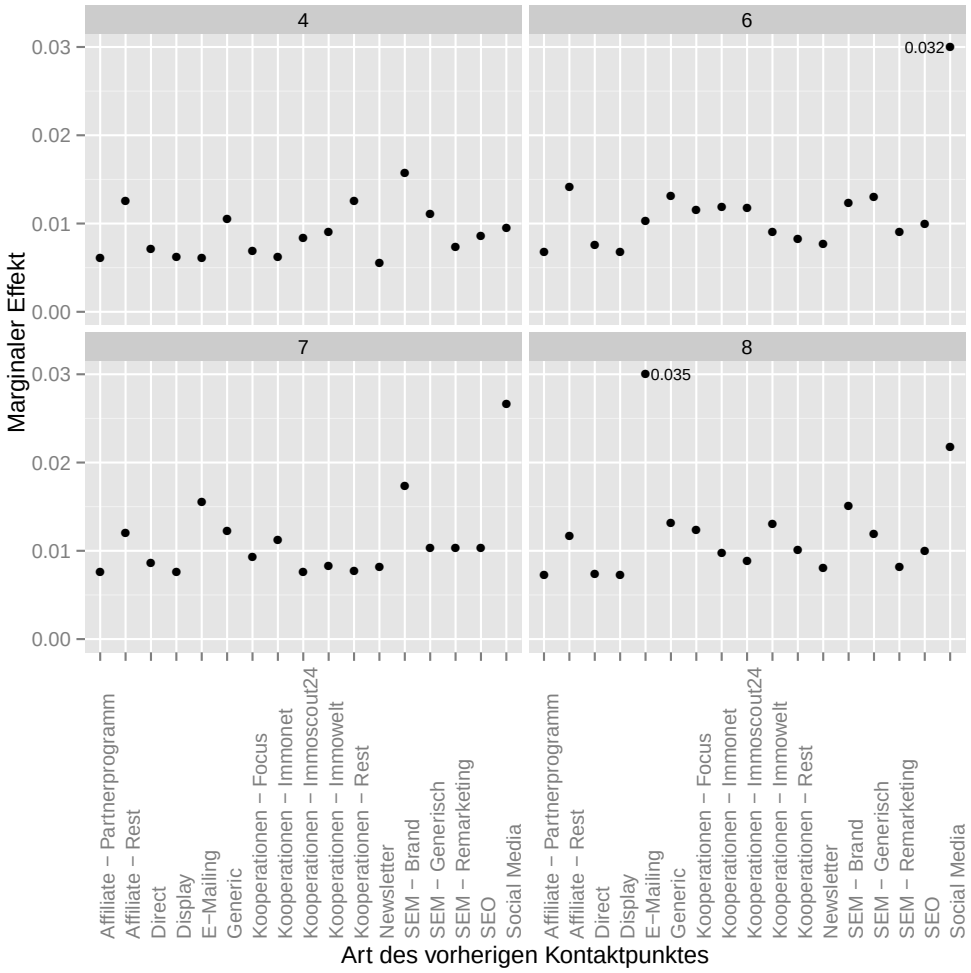

Abbildung 15: Marginaler Effekt des Features *CampaignLast* an den Positionen 4, 6, 7 und 8

Das Feature *TimeSinceFirst* ist an den Positionen 10 und 14 das wichtigste sowie an Position 11 und 12 das zweit- beziehungsweise drittwichtigste. *TimeSinceLast* ist an den Positionen 6, 12 und 13 das wichtigste Feature und an Position 8 das zweitwichtigste Feature. Die marginalen Effekte für diese Positionen sind in Abbilung 16 dargestellt, wobei diese an anderen Positionen ähnlich aussehen. Auf der x-Achse ist die Zeit in Tagen aufgetragen und auf der y-Achse die Marginalen Effekte. Für beide Features und alle Positionen steigen die marginalen Effekte mit der Zeit, wobei der Effekt bei *TimeSinceLast* etwas größer ist.
Dass die Konvertierungswahrscheinlichkeit bei zeitlich längeren Funnels höher ist, könnte man dadurch erklären, dass die Entscheidung für den Bau eines Hauses oder ähnlichem und das damit verbundene Ausfüllen eines Online-Antrages viel Zeit in Anspruch nimmt. Dass für die Zeit seit des vorherigen Kontaktes der selbe Effekt auftritt, könnte man ähnlich erklären. Bei der Entscheidung für eine Baufinanzierung sind viele Dinge zu

beachten. So erscheint es plausibel, dass sich interessierte Kunden eine längere Zeit auch offline mit dem Thema beschäftigen und später wieder online den Kontakt zur Interhyp AG suchen.

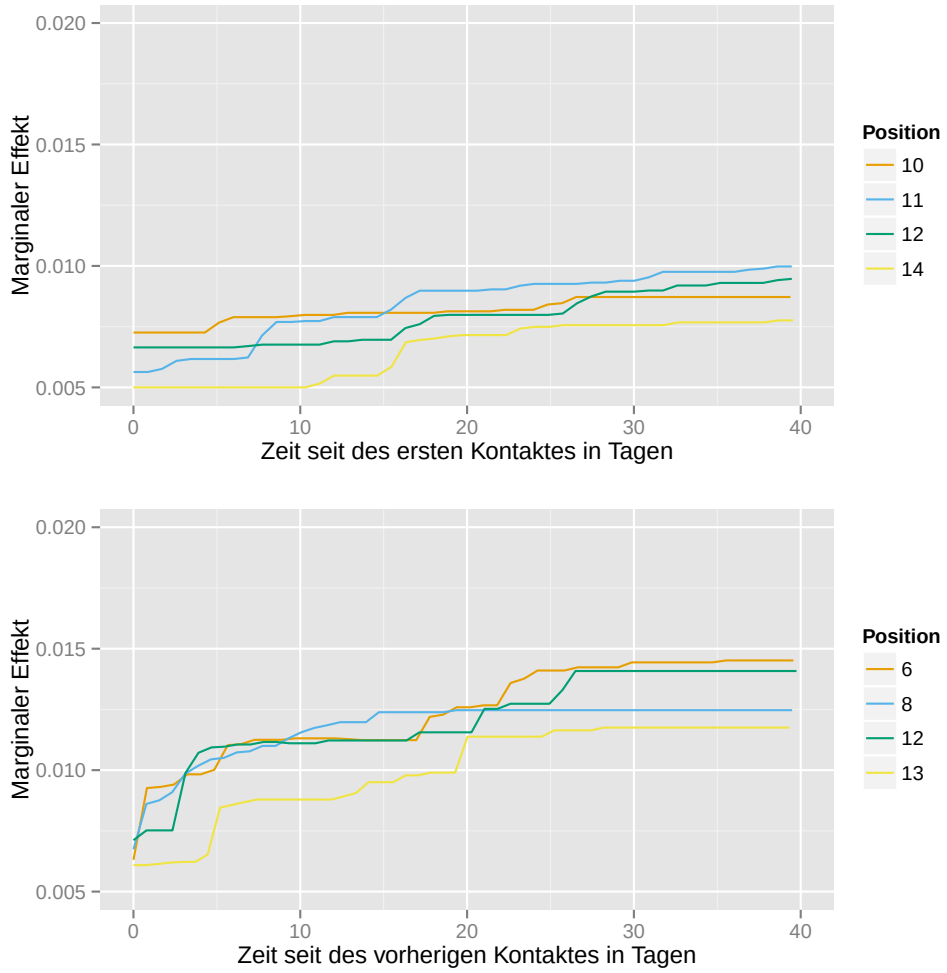

Abbildung 16: Marginaler Effekt des Features *TimeSinceFirst* an den Positionen 9, 10, 13 und 14 (oben) und des Features *TimeSinceLast* an den Positionen 6, 7, 8, 10 und 12 (unten)

Bis hierhin wurden die Ergebnisse des, auf den Trainingsdaten geschätzten, Modells vorgestellt. Nun soll das Modell auf die Testdaten angewendet werden, um die Prognosegüte zu beurteilen.
Wenn man die geschätzte Prognosefunktion auf die Testdaten anwendet, bekommt man für jede Position und jede ID aus den Testdaten die Hazardrate, das heißt die Wahrscheinlichkeit an einer bestimmten Position zu konvertieren. Diese sind sowohl für konvertierte als auch nicht-konvertierte Funnels sehr niedrig, da die absolute Anzahl an nicht-konvertierten Funnels deutlich überwiegt. Um die Prognosegüte des Modells näher zu betrachten wurde deshalb für jede Position eine ROC-Kurve berechnet. Diese sind in

Abbildung 17 für vier Positionen beispielhaft dargestellt.
An Position 1 ist dieser Unterschied noch nicht sehr deutlich. Dies kann daran liegen, dass hier nur die Features *Campaign*, *Hour* und *Weekday* einfließen und noch keine Informationen über vorherige Positionen vorliegen. Von dort steigen die Hazardraten an, wobei dieser Anstieg in den konvertierten Funnels deutlich stärker ist als in den nicht-konvertierten. Ab Position 6 werden die Konvertierungswahrscheinlichkeiten wieder geringer, wobei immer noch ein deutlicher Unterschied zwischen konvertiert und nicht-konvertiert zu erkennen ist.
Auf der x-Achse ist der Anteil der nicht-konvertierten Funnels, die als konvertiert vorhergesagt wurden aufgetragen und auf der y-Achse der Anteil der konvertierten Funnels, die auch als konvertiert vorhergesagt wurden. Je weiter die ROC-Kurve also oberhalb der roten Diagonalen liegt, desto besser kann das Modell zwischen konvertierten und nicht-konvertierten Funnels trennen. Für Position 3 liegt die Kurve für alle Punkte überhalb der Kurve von Position 1. Dies ist dadurch zu erklären, dass an Position 3 mehr Informationen zur Erstellung des Modells verwendet werden als an Position 1. Für Position 15 und 22 ist das Modell schlecht als an Position 3. Die Kurven für die späteren Positionen sind deutlich rauher. Dies ist darauf zurück zu führen, dass dort weniger Datenpunkte vorhanden sind. Durch das Einbringen des Offsets kann die Prognoseleistung des Modells allerdings halbwegs konstant gehalten werden.

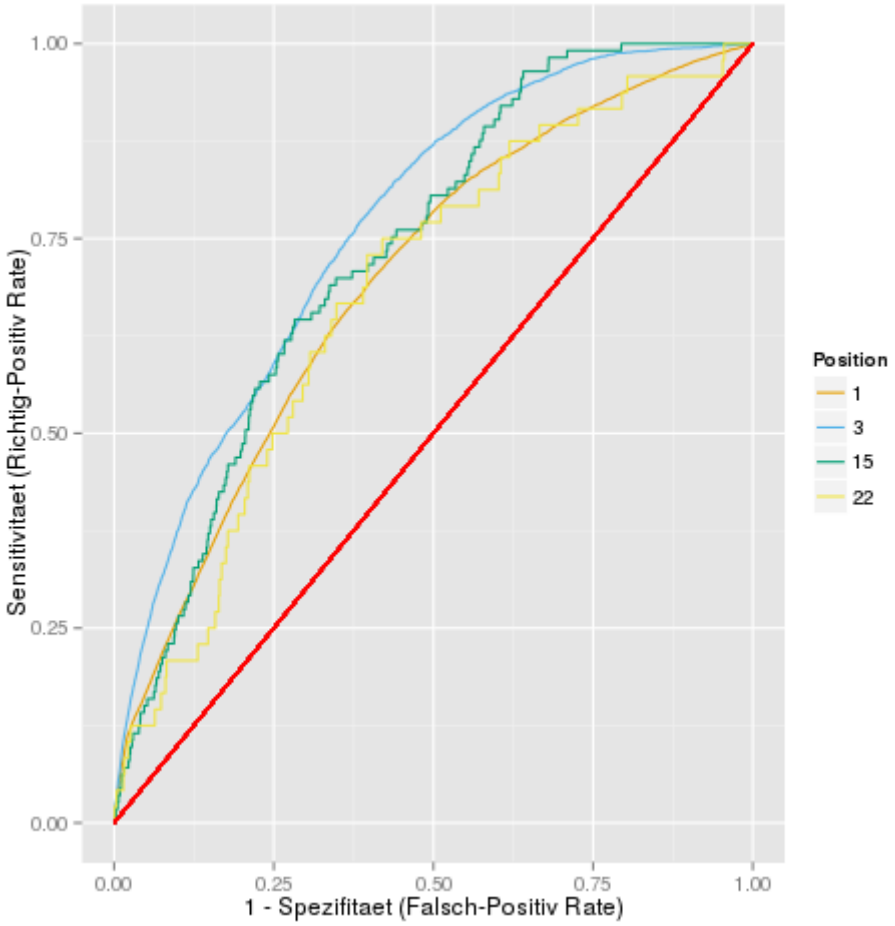

Abbildung 17: ROC-Kurve für die Positionen 1, 3, 15 und 22

Dies ist in Abbildung 18 noch deutlicher zu erkennen. Hier ist die Fläche unterhalb der ROC-Kurve (AUC) für jede Position abgebildet. Dieser Wert fällt zwischen Position 15 und 22 unter 0.75 ab, hält sich ansonsten aber relativ konstant auf 0.75. Das heißt die Wahrscheinlichkeit, dass bei einer konvertierten Beobachtung die Hazardrate größer ist als bei einer nicht-konvertierten Beobachtung ist positionsübergreifend circa 0.75.

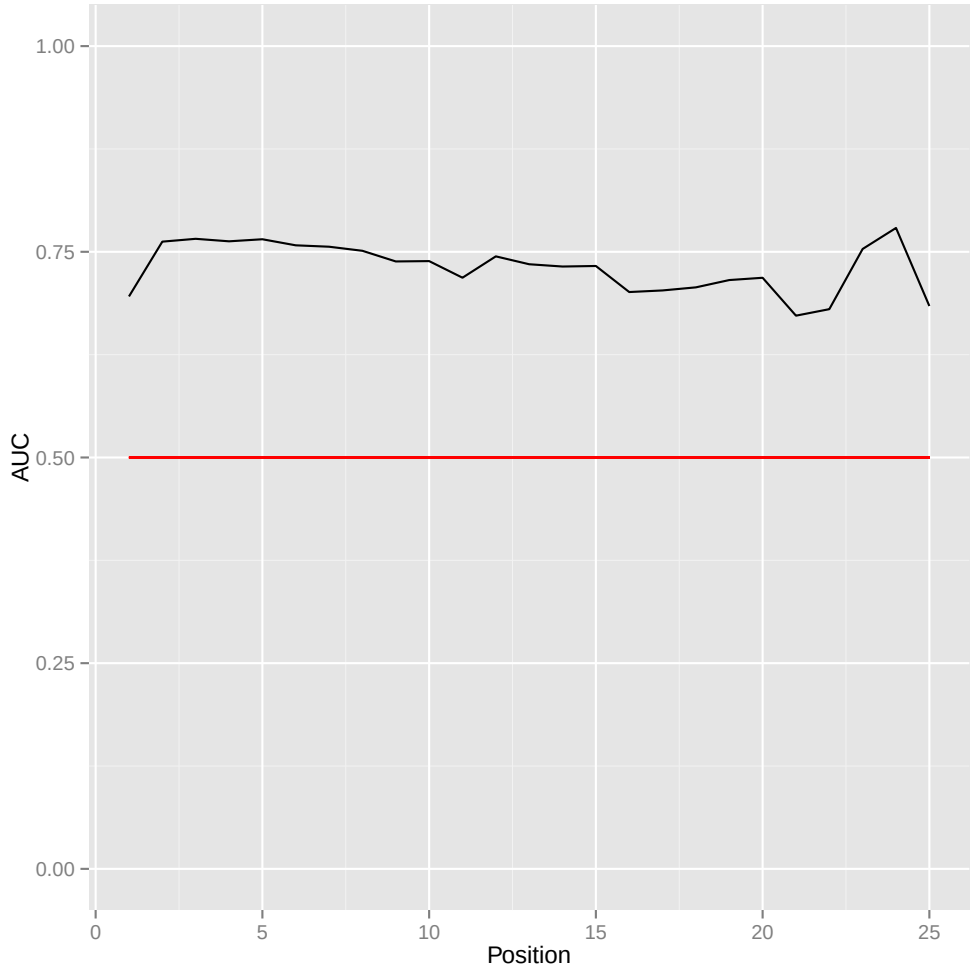

Abbildung 18: AUC für alle Positionen

7.2 Sequential Pattern Mining

Der in Kapitel 5 beschriebene Algorithmus wurde zunächst seperat auf alle konvertierten sowie nicht-konvertierten Funnels angewendet. Dabei wurden Funnels mit der Länge eins ausgeschlossen, da in diesen offentsichtlich keine interessanten Sequenzen enthalten sein können. Der minimale Support wurde auf 0.05 gesetzt, so dass nur Sequenzen, die in mindesten 5% aller Funnels vorkommen, als Ergebnis ausgegeben werden. In Abbildung 19 sind diese Sequenzen geplottet, wobei auf der x-Achse der Suppport aufgetragen ist und die Sequenzen anhand des Supports der Sequenzen in den konvertierten Funnels absteigend geordnet sind. An den Stellen, wo keine Balken geplottet sind, war der Support kleiner als 0.5. Die Namen der Kampagnen sind teilweise abgekürzt, um die Darstellung zu verbessern. *AffPar* steht für *Affiliate - Partnerprogramm*, *Dir* für *Direct* und *Dis* für *Display*. Außerdem ist zu beachten, dass die Sequenzen nicht exakt in der dargestellten Weise in den Funnels vorkommen müssen, sondern auch Abstände zwischen den aufein-

anderfolgenden Kampagnen erlaubt sind.
Der Support von Sequenzen der Länge eins, wie $<\{Dir\}>$ oder $<\{SEO\}>$, geben lediglich an, wie groß der Anteil der Funnels ist, die diese Kampagne mindestens einmal enthalten. Interessanter sind Sequenzen, die mindestens zwei Kampagnen enthalten. Hier fällt auf, dass Sequenzen mit wiederholtem *Direct*-Kontakt in den konvertierten Funnels stärker sind. Umgekehrt sind Sequenzen mit wiederholtem *Affiliate - Partnerprogramm*-Kontakt in den nicht-konvertierten Funnels stärker. Allerdings haben die Sequenzen insgesamt einen sehr geringen Support. Das liegt vor allem daran, dass die Daten zu großem Teil aus sehr kurzen Funnels bestehen (siehe Abbildung **FUNNEL LENGTH PLOT**).

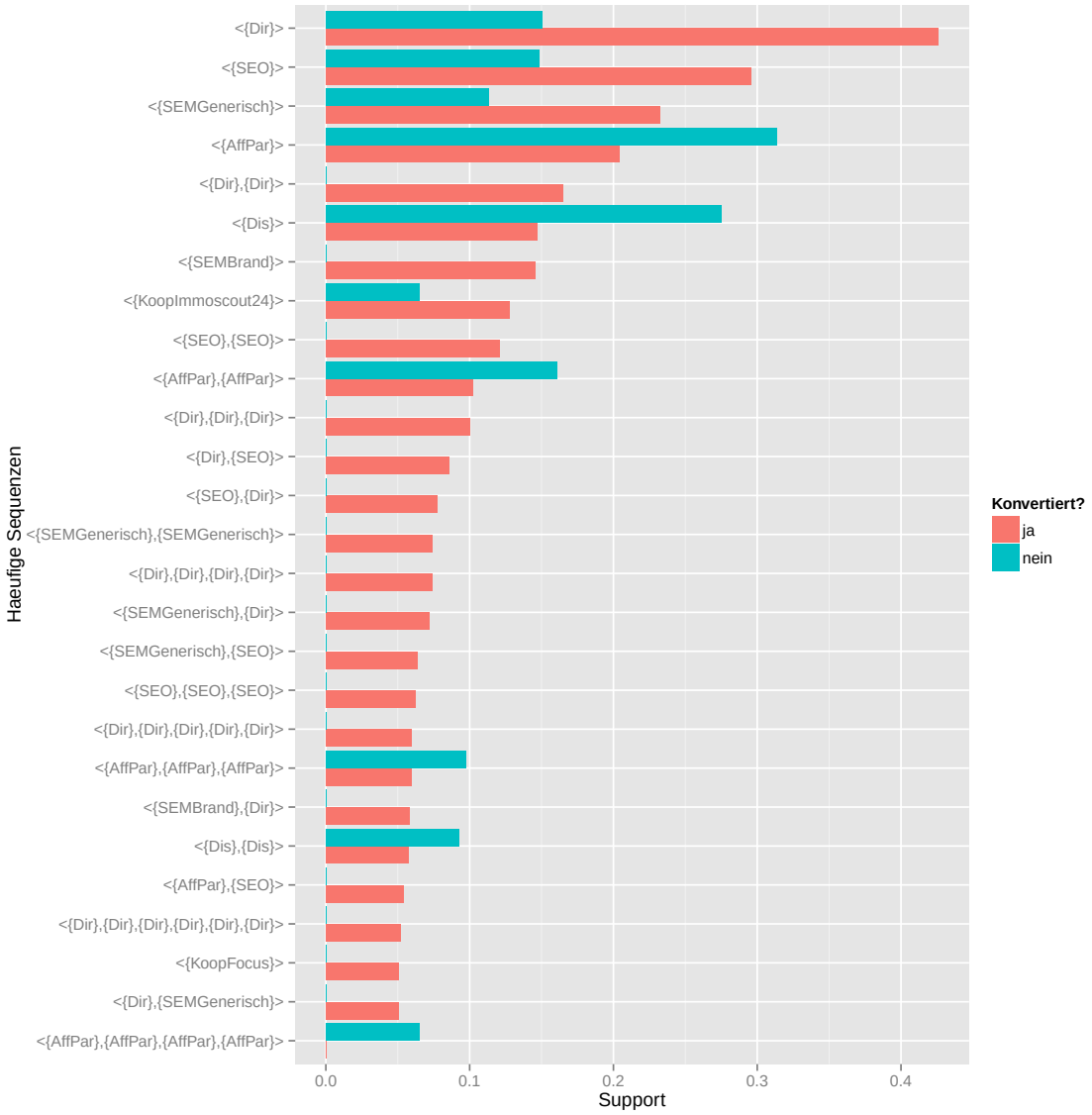

Abbildung 19: Häufige Sequenzen in den konvertierten und nicht-konvertierten Funnels

Deshalb wurde der SPADE-Algorithmus erneut seperat auf konvertierte und nicht-konvertierte Funnel angewendet, wobei dieses mal nur Funnels verwendet wurden, die eine Mindestlänge von 15 haben. Der minimale Support wurde auf 0.2 erhöht, um die Anzahl an häufigen Sequenzen zu beschränken. Die Ergebnisse sind in Abbildung 20 dargestellt.

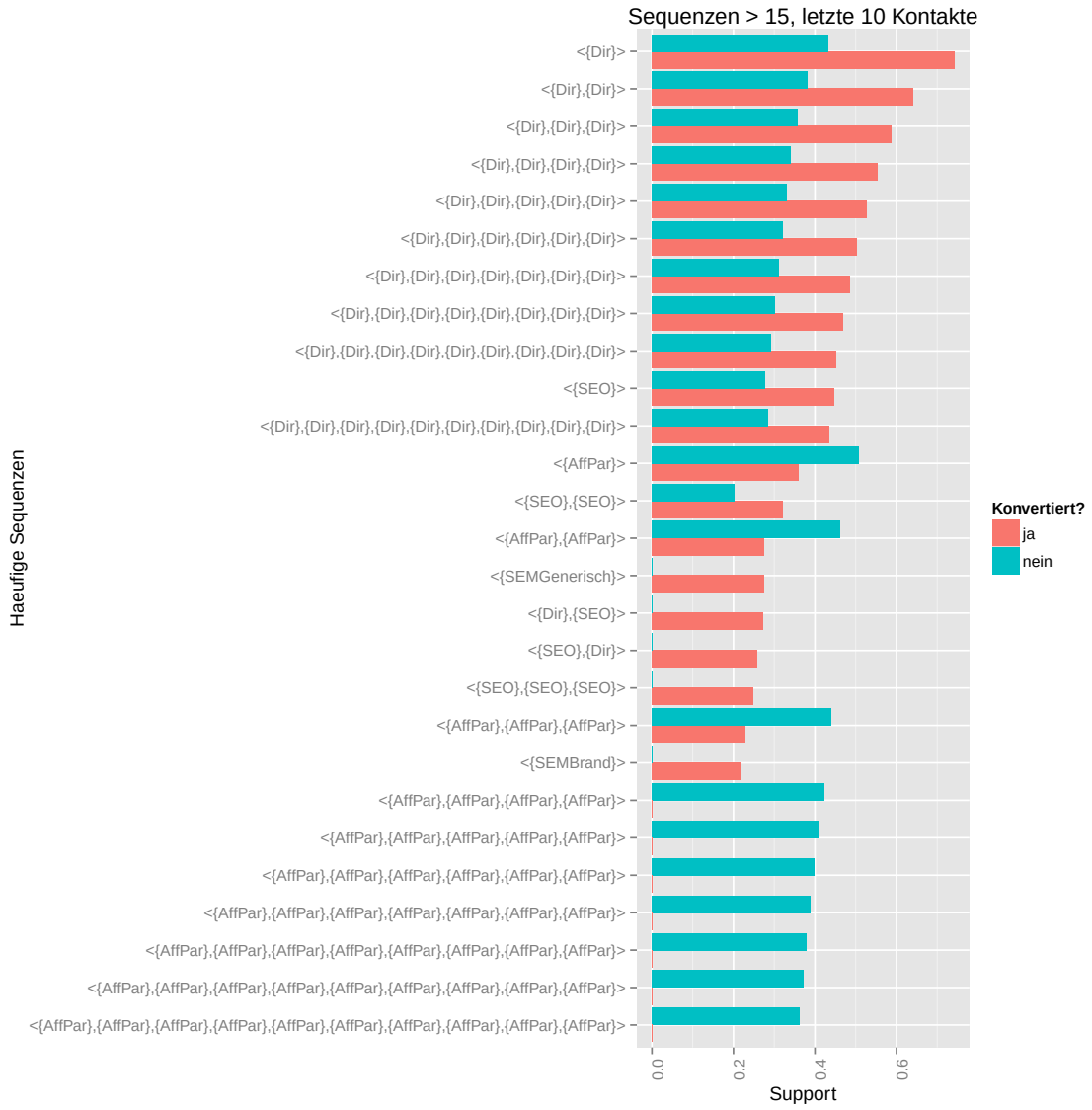

Abbildung 20: Häufige Sequenzen in den konvertierten und nicht-konvertierten Funnels mit mindestens 15 Kontakten

Die Sequenzen haben jetzt teilweise einen Support, der größer als 0.5 ist, das heißt mehr als die Hälfte der Funnels beinhalten diese Sequenz. Hier ist noch deutlicher zu erkennen, dass die *Direct*-Sequenzen in den konvertierten Funnels stärker sind als in den

nicht-konvertierten Funnels. Die *Affiliate - Partnerprogramm*-Sequenzen haben in den nicht-konvertierten Funnels einen Support von ungefähr 40% und in den konvertierten Funnels einen Support unter 20%.
GRAPHENTHEORETISCHES BLABLABLA

8 Zusammenfassung

9 Elektronischer Anhang

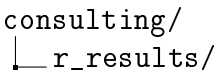

Abbildung 21: Verzeichnisstruktur des Projekts

Tabellenverzeichnis

Abbildungsverzeichnis

Literatur

[1] R. Agrawal & R. Srikant. „Mining sequential patterns“. In: *Proceedings of the 11th Conference on Data Engineering (ICDE'95)* (1995), S. 3–14.

[2] Revolution Analytics & Steve Weston. *doSNOW: Foreach parallel adaptor for the snow package*. R package version 1.0.12. 2014. URL: `http://CRAN.R-project.org/package=doSNOW`.

[3] Revolution Analytics & Steve Weston. *foreach: Foreach looping construct for R*. R package version 1.4.2. 2014. URL: `http://CRAN.R-project.org/package=foreach`.

[4] Christian Buchta, Michael Hahsler & with contributions from Daniel Diaz. *arulesSequences: Mining frequent sequences*. R package version 0.2-6. 2014. URL: `http://CRAN.R-project.org/package=arulesSequences`.

[5] M. El-Sayed, C. Ruiz & E. A. Rundensteiner. „FS-Miner: Efficient and incremental mining of frequent sequence patterns in web logs“. In: *Proceedings of the 6th Annual ACM International Workshop on Web Information and Data Management* (2004), S. 128–135.

[6] C. I. Ezeife, Y. Lu & Y. Liu. „PLWAP sequential mining: Open source code“. In: *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementation* (2005), S. 26–35.

[7] F. M. Facca & P. L. Lanzi. „Recent developments in web usage mining research“. In: *Proceedings of the 5th International Conference on Data Warehousing and Knowledge Discovery* (2003).

[8] J. H. Friedman, T. Hastie & R. Tibshirani. „Additive Logistic Regression: a Statistical View of Boosting“. In: *Annals of Statistics* 28.2 (2000), S. 337–374.

[9] J. H. Friedman. „Greedy Function Approximation: A Gradient Boosting Machine“. In: *Annals of Statistics* 29.5 (2001), S. 1189–1232.

[10] J. H. Friedman & B. E. Popescu. „Predictive Learning via Rule Ensembles“. In: *The Annals of Applied Statistics* 2.3 (2008), S. 916–954.

[11] B. Goethals. „Frequent set mining“. In: *The Data Mining and Knowledge Discovery Handbook* (2005), S. 377–397.

[12] R. Iváncsy & I. Vajk. „Frequent pattern mining in web log data“. In: *Acta Polytech. Hungarica* 3.1 (2006), S. 77–90.

[13] Y. Lu & C. I. Ezeife. „Position coded pre-order linked WAP-tree for web log sequential pattern minings“. In: *Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining* (2003), S. 337–349.

[14] Nizar R. Mabroukeh & C. I. Ezeife. „A Taxonomy of Sequential Pattern Mining Algorithms“. In: *ACM Computing Surveys* 43.1 (2010), 3:1–3:41.

[15] F. Masseglia, M. Teisseire & P. Poncelet. „Sequential pattern mining: A survey on issues and approaches“. In: *Encyclopedia of Data Warehousing and Mining* (2005), S. 1–14.

[16] Greg Ridgeway with contributions from others. *gbm: Generalized Boosted Regression Models*. R package version 2.1. 2013. URL: `http://CRAN.R-project.org/package=gbm`.

[17] J. Pei et al. „PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth“. In: *Proceedings of the International Conference on Data Engineering* (2001), S. 215–224.

[18] R Development Core Team. *R: A Language and Environment for Statistical Computing*. ISBN 3-900051-07-0. R Foundation for Statistical Computing. Vienna, Austria, 2012. URL: `http://www.R-project.org/`.

[19] S. Song, H. Hu & S. Jin. „HVSM: A new sequential pattern mining algorithm using bitmap representation“. In: *Advanced Data Mining and Applications* 3584 (2005), S. 455–463.

[20] J. Srivastava et al. „Web usage mining: Discovery and applications of usage patterns from Web data“. In: *ACM SIGKDD* 2.1 (2000), S. 12–23.

[21] J. Wang & J. Han. „BIDE: Efficient mining of frequent closed sequences“. In: *Proceedings of the 20th International Conference on Data Engineering* (2004), S. 79–90.

[22] M. J. Zaki. „SPADE: An efficient algorithm for mining frequent sequences“. In: *Mach. Learn.* 42 (2001), S. 31–60.