Master-Thesis

Studiengang Medizinische Informatik Master

Fakultät für Informatik

Hochschule Mannheim

26.11.2015

Durchgeführt bei der Firma Qudosoft GmbH & Co. KG, Karlsruhe

Betreuer: Prof. Ivo Wolf, Hochschule Mannheim

Zweitkorrektor: Jurgis Pods, Qudosoft GmbH & Co. KG

**Vergleich, Optimierung und Evaluation verschiedener statistischer Modelle zur Vorhersage der Wiederkaufwahrscheinlichkeit von Neukunden im Versandhandel.**

Nicholas Pastuovic

**Erklärung**

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

Karlsruhe, 26.11.2015

Nicholas Pastuovic

Inhalt

[1 Einleitung 1](#_Toc436292860)

[1.1 Motivation und Zielsetzung 1](#_Toc436292861)

[1.2 Aufteilung der Arbeit 2](#_Toc436292862)

[2 Grundlagen 3](#_Toc436292863)

[2.1 Klassifikationsverfahren 3](#_Toc436292864)

[2.1.1 Random Forests 3](#_Toc436292865)

[2.1.2 Support Vector Machine 3](#_Toc436292866)

[2.1.3 Logistische Regression 4](#_Toc436292867)

[2.1.4 Künstliche Neuronale Netze 4](#_Toc436292868)

[2.2 Maschinelles Lernen 5](#_Toc436292869)

[2.2.1 Überwachtes Lernen 5](#_Toc436292870)

[2.2.2 Unüberwachtes Lernen 5](#_Toc436292871)

[2.2.3 Bestärkendes Lernen 5](#_Toc436292872)

[2.3 Bewertung 5](#_Toc436292873)

[2.3.1 Cross Validation 6](#_Toc436292874)

[2.3.2 ROC-Kurve 6](#_Toc436292875)

[2.3.3 F-Score 6](#_Toc436292876)

[2.4 Verwendete Frameworks 6](#_Toc436292877)

[2.4.1 Pandas 6](#_Toc436292878)

[2.4.2 Scikit-learn 6](#_Toc436292879)

[3 Frameworkauswahl für Neuronale Netze 7](#_Toc436292880)

[3.1.1 PyBrain 7](#_Toc436292881)

[3.1.2 Tensorflow 7](#_Toc436292882)

[3.1.3 Lasagne 8](#_Toc436292883)

[3.1.4 Theano 8](#_Toc436292884)

[3.1.5 Fazit 8](#_Toc436292885)

[Tensorflow 8](#_Toc436292886)

[Theano 8](#_Toc436292887)

[Lasagne 8](#_Toc436292888)

[Pybrain 8](#_Toc436292889)

[5 Aktueller Stand der Entwicklung 9](#_Toc436292890)

[5.1 Erster Lösungsansatz 9](#_Toc436292891)

[5.2 Verwendeter Datensatz 10](#_Toc436292892)

[5.3 Ergebnis und Problematik 10](#_Toc436292893)

[6 Umsetzung 11](#_Toc436292894)

[6.1 Verwendung von Neuronalen Netzen 11](#_Toc436292895)

[6.1.1 Verwendung von SVM 11](#_Toc436292896)

[6.1.2 Verwendung von Logistischer Regression 11](#_Toc436292897)

[7 Evaluierung 12](#_Toc436292898)

[8 Fazit 13](#_Toc436292899)

[Abkürzungsverzeichnis iv](#_Toc436292900)

[Tabellenverzeichnis v](#_Toc436292901)

[Abbildungsverzeichnis vi](#_Toc436292902)

[Literaturverzeichnis vii](#_Toc436292903)

# Einleitung

## Motivation und Zielsetzung

In den Big Data Labs der Firma Qudosoft wird derzeit u.a. ein Vorhersagemodell auf anonymisierten Kundendaten eines großen klassischen Versandhändlers erstellt. Eine besonders interessante Fragestellung betrifft die Identifizierung von "One-time buyers" (OTBs): Kunden, die einmalig bestellen und nicht wiederkehren. Diese Kunden stellen für das Unternehmen ein Verlustgeschäft dar, da sie Werbekosten verursachen, ohne jedoch Nachfrage zu generieren. Gleichzeitig ist das Problem aus Sicht des maschinellen Lernens ein extrem schwieriges, da die Einordnung eines Neukunden als OTB oder Wiederkäufer lediglich anhand der Signatur seines Ersteinkaufs erfolgen muss. In einem ersten Klassifikationsversuch wurde eine Genauigkeit von 65% erreicht. Dafür wurde ein 250000 großer Datensatz mit je 40-200 Features, 2-5 Parameter pro Modell und Random Forest als Klassifikationsverfahren verwendet.

Es ist daher wünschenswert, dieses Vorhersageproblem methodologisch eingehender zu untersuchen. Konkret werden verschiedene statistische Modelle wie Logistische Regression, Support Vector Maschines (SVM) und Neuronale Netze auf den vorhandenen Daten trainiert. Mittels Kreuzvalidierungsverfahren (engl. Cross Validation) sollen optimale Parameter für jedes Modell gefunden werden. Dazu wird die vorhandene Hadoop-Infrastruktur genutzt, um diese aufwendigen Berechnungen parallelisiert durchführen zu können. Die dabei derzeit verwendeten Technologien umfassen Python, Scikit-learn, Pandas, Hive und Apache Spark.

Als Ziel sollen alle optimiert berechneten Modelle auf ungesehenen Daten hinsichtlich verschiedener Zielkriterien wie Genauigkeit, Precision, Recall und F-score ausgewertet und verglichen werden. Desweiteren soll eine Genauigkeit von 80% erreicht werden. Eine derzeit in der Entwicklung befindliches Web-Interface könnte in diesem Zuge zu einem komfortabel zu bedienenden Dashboard erweitert werden, dass das Fitten der Modelle, deren Optimierung per Cross Validation und den abschließenden Performance-Vergleich von einer einzigen Oberfläche aus ermöglicht.

## Aufteilung der Arbeit

In Kapitel 2 werden die Grundlagen verschiedener Klassifikations-Algorithmen, statistische Modelle und alle verwendeten Frameworks vorgestellt. In Kapitel 3 werden die verwendeten Daten und die charakteristischen Merkmale erläutert. Desweiteren wird auf den aktuellen Stand der Entwicklung eingegangen. Es wird genau beschrieben mit welcher prozentualen Genauigkeit eine Vorhersage getroffen wird.

Im vierten Kapitel wird die OTB-Problematik an verschiedenen Klassifikationsverfahren angewandt. Mithilfe von Kreuzvalidierungsverfahren sollen die einzelnen Parameter der Verfahren bestmöglich eingestellt werden. In Kapitel 5 werden die Verfahren mithilfe von einem Dashboard untereinander verglichen. Hierzu werden z.B. Receiver Operating-Characteristic-Kurven (ROC) zur Beurteilung der Klassifikatoren verwendet.

Im letzten Kapitel werden die gewonnenen Erfahrungen diskutiert und daraus die entsprechenden Schlussfolgerungen gezogen.

## 

# Grundlagen

## Klassifikationsverfahren

Klassifikationsverfahren beschreiben den Prozess gegebene Objekte in Klassen einzuteilen. Zu welcher Klasse ein Objekt gehört wird über dessen Merkmale entschieden. Im Weiteren werden verschiedene Verfahren zur Lösung des Problems vorgestellt.

### Random Forests

Random Forests ist ein Klassifikationsverfahren, welches zur Lösung ein Ensemble von Entscheidungsbäumen bildet.

#### Bagging

### Support Vector Machine

Support Vector Machine (SVM) ist ein supervised learning Verfahren, welches versucht eine Menge von Daten in Klassen zu unterteilen. Dabei wird eine Hyperebene, die als Trennlinie dient, so gelegt, dass sie die verschiedenen Klassen optimal trennt.

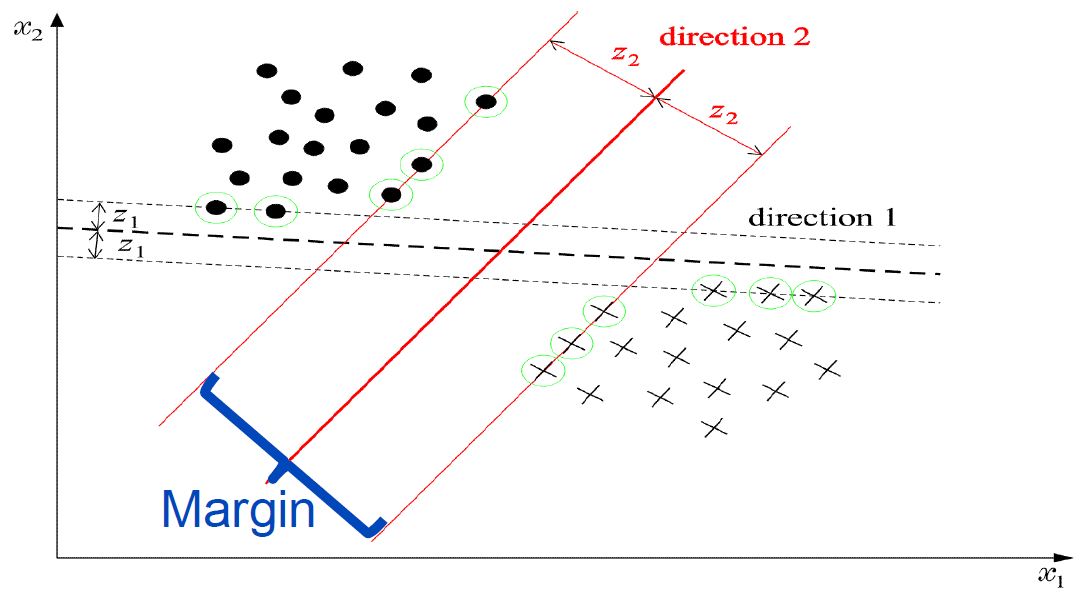


Abbildung : Support Vector Machine [Buchquelle]

Wie man in Abbildung 1 sieht, gibt es mehrere Möglichkeiten eine Trennlinie zwischen zwei Klassen zu legen. Optimal wird die Hyperebene so gelegt, dass der Abstand der zwischen den Klassen maximal ist. Liegt sie zu nah an einer Klasse, ist die Klassifizierung sehr anfällig und eine korrekte Verallgemeinerung wird dadurch nicht gewährleistet. Die Objekte die der Hyperebene am nächsten liegen, werden Support Vectors genannt. In Abbildung 1 sind sie grün umkreist zu sehen. Um einen eine Optimale Trennung zu erreichen, wird der Abstand zwischen den Support Vectors der verschiedenen Klassen berechnet und die Hyperebene daraufhin so gelegt, dass der Abstand der Support Vectors zur Hyperebene möglichst groß ist.

#### Nicht linear trennbare Hyperebenen

### Logistische Regression

### Künstliche Neuronale Netze

Künstliche Neuronale Netze (KNN) sind große informationsverarbeitende Systeme, die als Vorbild das menschliche Gehirn haben. Sie bestehen aus mehreren Einheiten bzw. Neuronen die untereinander Verbunden sind.

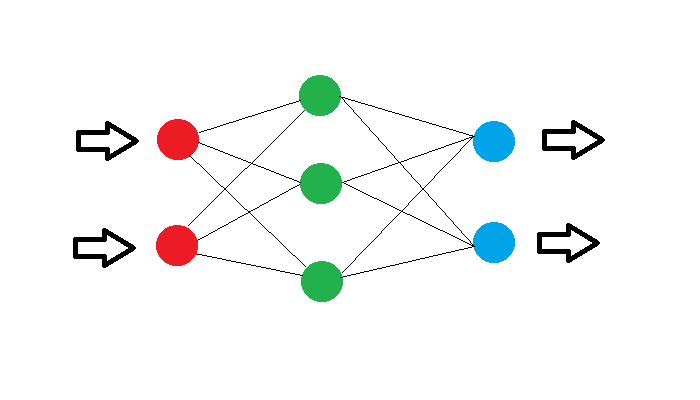


Abbildung : Neuronales Netz

## Maschinelles Lernen

### Überwachtes Lernen

Beim Überwachten Lernen (engl. supervised learning) lernt der Algorithmus anhand einer Menge von Trainingsdaten, die aus Ein- und Ausgaben bestehen, Gesetzmäßigkeiten. Ein „Lehrer“ gibt dem Algorithmus die korrekten Ausgaben für jede Eingabe vor. Nach dem Lernvorgang ist das System in der Lage Gesetzmäßigkeiten nachzubilden.

Diese Arbeit wird ausschließlich Verfahren für das überwachte Lernen behandeln.

### Unüberwachtes Lernen

Beim Unüberwachten Lernen (engl. unsupervised learning) ist kein „Lehrer“ anwesend der die korrekten Antworten vorgibt. Der Algorithmus gruppiert selbstständig die Objekte anhand ihrer Merkmale in Klassen.

### Bestärkendes Lernen

Beim Bestärkendes Lernen (engl. Reinforcement Learning) ist eine Menge von Datensätzen vorhanden, jedoch gibt es keine Vorgabe der Zielwerte. Der Algorithmus versucht eine Gesetzmäßigkeit herzustellen und ein Bewerter bestimmt im Anschluss wie gut das Ergebnis ist. Anhand der Bewertung verfeinert der Algorithmus sein Ergebnis.

## Bewertung

Im Folgenden werden verschiedene Verfahren zur Bewertung der verwendeten Modelle beschrieben.

### Cross Validation

### ROC-Kurve

Mit ROC-Kurven lassen sich statistische Modelle Optimieren und Bewerten.

### F-Score

## Verwendete Frameworks

In diesem Unterkapitel werden alle, für diese Arbeit relevanten Frameworks vorgestellt. Es wird beschrieben welche Anwendungsfälle die einzelnen Frameworks abdecken.

### Pandas

### Scikit-learn

Das

# Bibliothekauswahl für Neuronale Netze

In diesem Kapitel wird die Auswahl der Bibliotheken zur Realisierung des Neuronalen Netz diskutiert. Es werden die gängigsten Bibliotheken genannt, kurze Code-Beispiele erläutert und auf die Vor- und Nachteile eingegangen.

### PyBrain

PyBrain ist eine Deep-Learning-Bibliothek für Python, welche ausschließlich Modellierungen Neuronaler Netze realisiert. Das Ziel der Bibliothek ist es möglichst einfach und flexibel Neuronale Netze zu bilden und einzusetzen.

1. n = buildNetwork(2,6,3, outclass=SoftmaxLayer )
3. t = BackpropTrainer(n, dataset=trndata,learningrate=0.01, momentum=0.1)
4. t.trainUntilConvergence(maxEpochs=200)
6. activatedNetwork = n.activateOnDataset(evaldata)

Listing : Erstellung eines Netzes mit einem Hidden-Layer

Wie in Listing 1 zu sehen, lässt sich unkompliziert ein Netz mit der *buildNetwork()*-Klasse erzeugen. Anhand der Übergabeparameter können Größe, Aktivierungsfunktion der einzelnen Layer und Struktur des Netzes definiert werden. Nachdem das Netz erstellt wurde kann mithilfe der *BackpropTrainer*-Klasse die Lernregel, Lernrate oder auch das Momentum-Term zur Generalisierung angegeben werden. Ist der Trainer eingestellt, kann das eigentliche Lernen beginnen. Mit der *trainUnitlConvergence()*-Methode wird der Lernvorgang gestartet. Es kann die Anzahl der Epochen sowohl die verwendeten Daten bestimmt werden. Ist das Lernen beendet kann mit der *activateOnDataset()*-Methode auf ungesehene Daten eine Vorhersage getroffen werden. Die Methoden liefert die Zugehörigkeit einer Klasse als Wahrscheinlichkeit zurück.

### Tensorflow

Tensorflow ist eine Deep-Learning-Bibliothek die von Google entwickelt wurde. Mit der Bibliothek lassen sich numerische Berechnungen als Datenflussgraphen realisieren. Wie man in Abbildung 2 sieht, sind Knoten mathematische Operationen und entlang der Kanten fließen Daten als Tensoren.

1. x = tf.placeholder("float", [None, n\_input])
2. y = tf.placeholder("float", [None, n\_classes])
4. weights = {
5. 'h1': tf.Variable(tf.random\_normal([size\_input, size\_hidden])),
6. 'out': tf.Variable(tf.random\_normal([size\_hidden, n\_classes]))
7. }
8. biases = {
9. 'b1': tf.Variable(tf.random\_normal([size\_hidden])),
10. 'out': tf.Variable(tf.random\_normal([size\_classes]))
11. }
13. layer = tf.nn.relu(tf.add(tf.matmul(x, \_weights['h1']), \_biases['b1']))
14. pred = tf.matmul(layer, weights['out']) + biases['out']
16. cost = tf.reduce\_mean(tf.nn.softmax\_cross\_entropy\_with\_logits(pred, y))
17. optimizer = tf.train.AdamOptimizer(learning\_rate=0.01).minimize(cost)
19. init = tf.initialize\_all\_variables()
21. with tf.Session() as sess:
22. sess.run(init)
23. **for** epoch **in** range(training\_epochs):
24. avg\_cost = 0.
25. **for** i **in** range(total\_batch):
26. sess.run(optimizer, feed\_dict={x: data, y: label})

Listing : Erstellen eines Netzes in Tensorflow

Um ein Netz mit Tensorflow zu erstellen müssen, wie in Listing 2 zu sehen, zuerst *Placeholder* deklariert werden, welche die In- und Output Schichten repräsentieren. Sind die Schichten mit ihrer jeweiligen Größe initialisiert, müssen die Gewichte und die Bias als *Variable* definiert werden. *Variablen* sind als feste Werte innerhalb des Datenflussgraphen. Um die grundlegende Struktur des Netzes zu vervollständigen müssen die Schichten, Gewichte, Bias und Aktivierungsfunktionen festgelegt werden. Tensorflow bietet eine Vielzahl an Aktivierungsfunktionen. --- Nachdem alle Knoten und Aktivierungsfunktionen verbunden sind, muss das Netz durch die *initialize\_all\_variables()*-Methode initialisiert werden. Danach kann der Datenflussgraph ausgeführt werden. Hierfür wird für jeden Graph eine *Session* erstellt, welche den Graphen, die dazugehörigen Daten und Optimierungsfunktion als Übergabeparameter entgegen nimmt. Um die *Placeholder* mit richtigen Daten zu „füttern“ wird der *run()-*Methode ein *feed\_dict* mit den Daten und der entsprechenden Zielvariable übergeben.

### Theano

### Lasagne

### Fazit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Tensorflow | Theano | Lasagne | Pybrain |
| Einfachheit |  |  |  |  |
| Dokumentation |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Tabelle : Bewertung der Bibliotheken

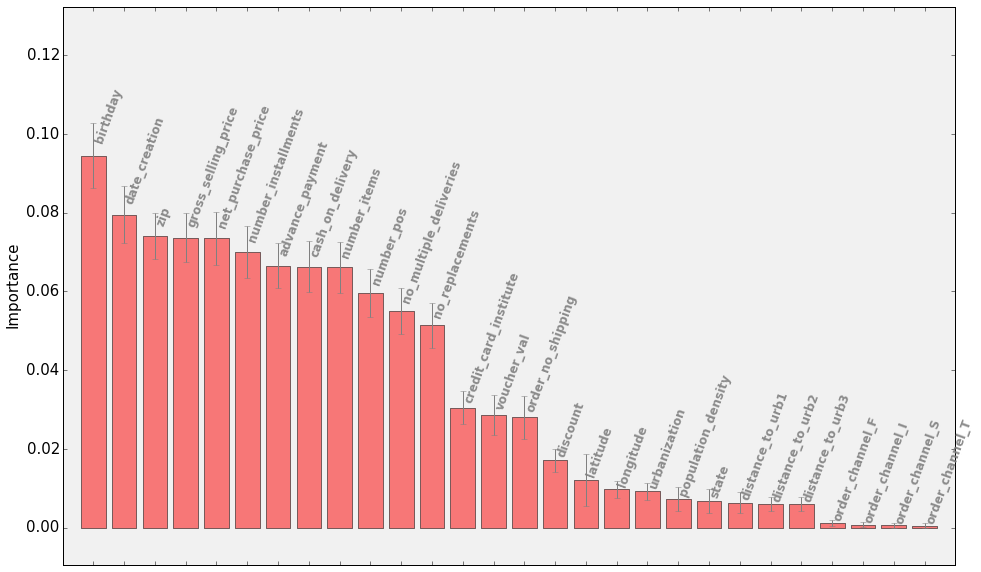
# Aktueller Stand der Entwicklung

In diesem Kapitel wird der aktuelle Stand der Entwicklung beschrieben. Es wird gezeigt welches Klassifizierungsverfahren zur Lösung der OTB-Problematik verwendet wurde. Genauer werden die verwendeten Parameter und Merkmale beschrieben, sowie den vorhandenen Datensatz. Zum Schluss des Kapitels wird die momentane Genauigkeit der Klassifizierung mittels einer ROC-Kurve visuell dargestellt.

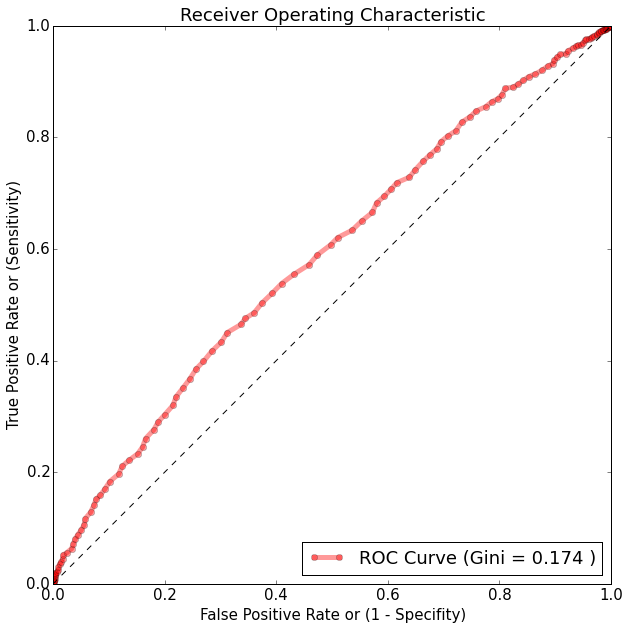
## Erster Lösungsansatz

Für die Lösung der OTB-Problematik wird als Programmierungssprache Python verwendet, da sie viele statistische Open-Source Bibliotheken frei zu Verfügung stellt. Als Klassifikationsverfahren wird der Random Forests Algorithmus gewählt. Ausschlaggebend für die Verwendung des Verfahrens ist seine Parallelisierbarkeit der Evaluierung, seine Effizienz in der Trainingszeit hinsichtlich der größer der Datenmenge.  
Zur Realisierung des Problems wird das Python-Paket „Scikit-learn„ verwendet, welche eine Implementierung des Random Forests bereitstellt. Zur Initialisierung des Modells wird eine Größe von 200 Bäumen gewählt. Die Anzahl der Merkmale für das „Feature-Bagging“, wird aus der Wurzel der maximalen Anzahl der vorhandenen Merkmale berechnet.

## Verwendeter Datensatz



## Ergebnis und Problematik



# Umsetzung

In Kapitel 4 wird die OTB-Problematik auf verschiedene Klassifizierungsverfahren angewandt. Es wird beschrieben wie die einzelnen Verfahren aufgebaut und parametrisiert werden. Mittels Cross-Validation sollen die Verfahren überprüft und optimiert werden.

## Verwendung von Neuronalen Netzen

### Verwendung von SVM

### Verwendung von Logistischer Regression

# Evaluierung

# Fazit

Abkürzungsverzeichnis

**ABK** Abkürzung

**OTB** One-time-buyers

**SVM** Support Vector Maschines

**KNN** Künstliche Neuronale Netze

**ROC** Receiver Operating Characteristic

Tabellenverzeichnis

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

Abbildungsverzeichnis

[Abbildung 1: Support Vector Machine [Buchquelle] 2](#_Toc433632597)

Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | M. Kornmeier, Wissenschaftlich schreiben leicht, 4. Hrsg., UTB, 2011. |