



Company Project # 2015-1428

Martin Nicklas Jørgensen `tzk173@alumni.ku.dk`

User Behavior Analysis Using Decision Trees

Supervisor: Mikkel Rønne Jakobsen `mikkellrj@di.ku.dk`

January 11, 2016

Abstract

Write something sensible, requirements is on the intranet page.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec a diam lectus. Sed sit amet ipsum mauris. Maecenas congue ligula ac quam viverra nec consectetur ante hendrerit. Donec et mollis dolor. Praesent et diam eget libero egestas mattis sit amet vitae augue. Nam tincidunt congue enim, ut porta lorem lacinia consectetur. Donec ut libero sed arcu vehicula ultricies a non tortor. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean ut gravida lorem. Ut turpis felis, pulvinar a semper sed, adipiscing id dolor. Pellentesque auctor nisi id magna consequat sagittis. Curabitur dapibus enim sit amet elit pharetra tincidunt feugiat nisl imperdiet. Ut convallis libero in urna ultrices accumsan. Donec sed odio eros. Donec viverra mi quis quam pulvinar at malesuada arcu rhoncus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. In rutrum accumsan ultricies. Mauris vitae nisi at sem facilisis semper ac in est.

Contents

1	Introduction	2
2	SimpleSite ApS	2
2.1	Product	2
2.2	Departments	3
2.3	Company Structure	3
3	Problem Description	5
3.1	Requirements	5
3.2	Success Criteria	5
4	Problem Analysis	6
4.1	Available Datasets	6
4.2	Dataset Pruning	7
4.3	Tree Type	8
4.3.1	Plotting	8
4.3.2	Model Precision	8
4.3.3	Choosing a Model	10
4.4	Model Analysis	10
4.4.1	Dataset Features	10
4.4.2	Formula and Tree Depth	10
5	Results	10
6	Competencies and Methods	10
7	Conclusion	10

1 Introduction

2 SimpleSite ApS

SimpleSite is a danish company that was founded as *Elk Consulting ApS* in 2001 by Morten and Jacob Elk [3]. In 2003 the company changed its name to *123Hjemmeside ApS* in order to reflect their product they delivered. Over the years as 123hjemmeside expanded into new countries the name was translated into the local equivalent for each version of the site. In 2014 123hjemmeside adopted its biname *SimpleSite ApS* as its new primary name in order to unify all its brands under a single name.

The information in this section is gathered from [3], interviews and from having worked in the operations department for 5 years.

2.1 Product

SimpleSite produce and maintain an inhouse website Content Management System (CMS) and operate a hosting location where this CMS runs. Customers can register for a free account allowing them to host a website that is edited through the CMS. The free accounts are under restrictions on number of pages, images and videos that can be added to the site. Customers can then change to a paid subscription which allows them to have more pages, images and videos on their website.

SimpleSite also offer additional services to their customers that can be bought for an extra fee if you are already a paying customer, this includes the ability to have a domain attached to your website, a webshop as well as more additional pages, images and videos.

The product is hosted partially on SimpleSites own hardware in an offsite location, and using a number of cloud services to provide faster response times for certain data types.

2.2 Departments

The employees work from three different offices located in Denmark, Bulgaria and Serbia, but also have people working from other locations around the world.

- **Administration** - Situated at the ground floor of the Copenhagen office, this department contains the HR functions as well as finance.
- **Sales** - Also on the ground floor of the Copenhagen office, sales consists of full time employees and student helpers. The primary task is to sell the product, currently through localized ad management.
- **Product & Communication** - The department sits on the upper floor of the Copenhagen office and is responsible for planning new features in cooperation with the developers as well as manage content on SimpleSites own websites. P & C also manages communication such as newsletters and localized ad texts.
- **In-House Development** - Sitting next to P & C on the upper floor the developers are responsible for implementing new features as well as maintenance, analytics and bug fixing of the product.
- **Operations** - Daily operations are handled primarily by the company CTO, Thomas, as well as 2 part time students. Operations sits on the upper floor in Copenhagen next to the developers.
- **Support** - Since the product is offered in several languages, a supporter is hired per language in a part time basis. All supporters work from home but get together once a month for a status meeting and to make sure new knowledge is shared and that relevant information can be given from the regular departments.
- **Remote Dev 1** - A small number of developers work in Bulgaria and have their own office in the Sofia, they are offered machines in the Copenhagen office they can VPN into and use for work. This allows the operations department to work from Copenhagen and still service the remote developers.
- **Remote Dev 2** - A number of developers work from Serbia, like the remote developers in Bulgaria, they also VPN into the Copenhagen office and work on machines maintained by the regular operations employees.
- **Miscellaneous** - SimpleSite also occasionally employs external specialists or consultants. Depending on the need, they will either work in the Copenhagen office or from some remote location using VPN.

2.3 Company Structure

Figure 1 shows the overall organization of SimpleSite, for simplicity I have merged the remote development offices into the “regular” development department since they occupy the same space in the organization, the only difference is the geographical separation.

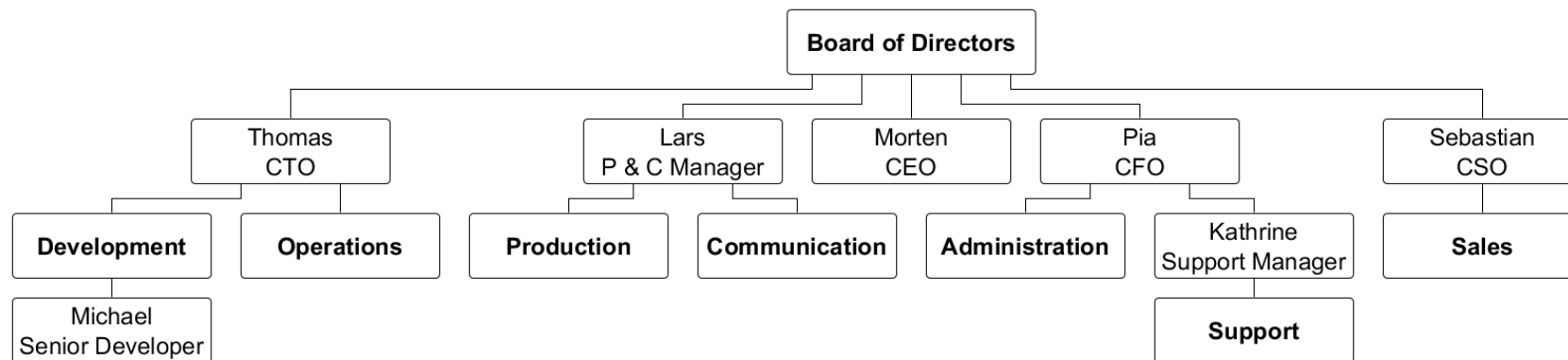


Figure 1: Organizational diagram for SimpleSite ApS, text in bold represents departments.

3 Problem Description

Before 2014 123Hjemmeside offered paid subscription that allowed customers to make and host a website using the provided CMS. Customers could get a time- limited trial but would have to buy a subscription afterwards. If a customer stopped paying their subscription or did not wish to continue after their trial, their website would be kept offline for a short while, but would be deleted afterwards.

When 123hjemmeside changed its name to SimpleSite in 2014 it also changed its business model. They went from being purely paid to allowing everyone to create as many websites as they wished for free, with no time limitation. The free subscriptions give access to most functions of the CMS, excluding things like personal domains, webshops and puts a limit on the number of images a website can contain. In order to use these features a customer will have to sign up for a paid subscription. This model of free and premium products is known as freemium ([1, p. 6] and [5, p. 1]).

After the change of model SimpleSite could see a huge increase in the number of free websites being created, but not a corresponding increase in paid subscriptions. They came up with the idea of recording how customers use the site and use the data to build decision trees ([2] and [6]). The decision trees should not only be to be able to classify customers based on behaviour, but also to let SimpleSite learn what events in the users life cycle that increase the chance of them becoming paying customers.

Initially SimpleSite is interested in exploring if there is any differences between the users that still have logins 15 or more days after they are created (SimpleSite calls these customers *retained*) and customers that simply make a site and leave or forget about it. This mapping of important “actions” that some customers take is supposed to be used later in a mail procedure that will automatically send out emails to new customers prompting them to perform these actions that are known to cause customers to be retained.

3.1 Requirements

SimpleSite have already created some code in R¹ so any continued work should be done in the R language. I have also been added to a Github repository containing analytics code and a preferred structure and coding style that should be adhered to.

Two datasets are also available containing different information gathered from the live system. These two datasets should form the basis for all the dataanalysis performed. An additional dataset is being constructed during the project, but due to the time it takes to populate it with enough observations to be meaningful it might not be finished before the project is over.

3.2 Success Criteria

The project have 4 success criteria (that should all be fulfilled to some degree):

1. A decision tree model for classifying customers created.
2. New knowledge about customer lifecycles acquired from the model, in particular, do *retained* customers have something in common.
3. A prototype R script that can automatically build/create the model instance from new customer data should be created.
4. A method for using the model should be designed or reasoned about.

¹<https://www.r-project.org/>

4 Problem Analysis

4.1 Available Datasets

The basis for the analysis is 2 datasets created by SimpleSite: **EngagementData** & **CustomerJourney**. Table 1 and Table 2 names the features of each dataset and what they represent. Both datasets contain users created between September 1st 2015 and September 30 2015. The data is recorded for all users afterwards as well, but the larger dataset also increases memory consumption, and makes it unwieldy.

This window can be expanded as long as the user it at least a month old.

Attribute Name	Attribute Data
<i>islogins1</i>	Bool, true if: one or more logins for the user.
<i>islogins2</i>	Bool, true if: two or more logins for the user.
<i>islogins3</i>	Bool, true if: three or more logins for the user.
<i>islogins4</i>	Bool, true if: four or more logins for the user.
<i>isedit30m</i>	Bool, true if: User edited site within 30 minutes of creation.
<i>isedit24h</i>	Bool, true if: User edited site within 24 hours of creation (excluding the first 30 minutes).
<i>isaddpage30m</i>	Bool, true if: User added a new page within 30 minutes of creation.
<i>isaddpage24h</i>	Bool, true if: User added a new page within 24 hours of creation (excluding the first 30 minutes).
<i>isimgupload30m</i>	Bool, true if: User uploaded their own image within 30 minutes of creation.
<i>isimgupload24h</i>	Bool, true if: User uploaded their own image within 24 hours of creation (excluding the first 30 minutes).
<i>iseditdesign30m</i>	Bool, true if: User edited site within 30 minutes of creation.
<i>iseditdesign24h</i>	Bool, true if: User edited site within 24 hours of creation (excluding the first 30 minutes).
<i>customerid</i>	Integer value with the customers unique ID.
<i>marketname</i>	String with the market the user came from (US, TR, DK etc.)
<i>siteverkey</i>	String with what version of the site the user is created in (US, TR, DK etc.)
<i>ispayer</i>	Bool, true if: The customer have a paid subscription.
<i>culturekey</i>	String with language information for the site (en-US, fr-FR etc.)

Table 1: Features found in the **EngagementData** dataset.

Attribute Name	Attribute Data
<i>customerid</i>	Integer value with the customers unique ID.
<i>logins14</i>	Integer, number of times the customer logged in the first 14 days (week 1-2 after creation).
<i>logisnw2w4</i>	Integer, number of times the customer logged in in week 3-4 after creation.
<i>edits14</i>	Integer, number of times the customer edited a page within the first 14 days.
<i>iscjtrial</i>	Bool, true if: Always true, everyone starts as a trial.
<i>iscjonboarded</i>	Bool, true if: $\text{edits14} \geq 1$.
<i>iscjactivated</i>	Bool, true if: $\text{edits14} \geq 3$.
<i>iscjengaged</i>	Bool, true if: $\text{edits14} \geq 6$ and $\text{logins14} \geq 2$.
<i>iscjinvested</i>	Bool, true if: $\text{edits14} \geq 15$ and $\text{logins14} \geq 6$.
<i>iscjretained</i>	Bool, true if: $\text{logisnw2w4} \geq 1$.
<i>isimgupload1d</i>	Bool, true if: Customer uploaded an image within the first 24 hours of being created.
<i>iseditdesign1d</i>	Bool, true if: Customer edited the design within the first 24 hours of being created.
<i>isaddpage1d</i>	Bool, true if: Customer added a new page within the first 24 hours of being created.
<i>isedit1d</i>	Bool, true if: Customer edited a page within the first 24 hours of being created.

Table 2: Features found in the **CustomerJourney** dataset.

4.2 Dataset Pruning

The initial goal is to find customers who are retained (*iscjretained* = **True**) and see if there is some pattern, that Simplesite can use to try to guide other customers down in order to increase the number of retained customers. With this in mind there is some attributes of the datasets that will not be helpful, either because they cannot be controlled/changed, or because they do not make sense. The following is a list of attributes removed from the **EngagementData** dataset during work, along with the reason for the removal.

- *islogins1* : Removed since it is always true for all customers.
- *islogins2* : Removed because the definition of a retained customer requires one or more logins, so this must always be true.
- *islogins3* : Same as *islogins2*.
- *islogins4* : Same as *islogins2*.
- *marketname* : Removed since we are unable to get a customer from a different market, we are interested in variable we can change for each customer.
- *siteverkey* : Same as *marketname*.
- *ispayer* : Removed because it is an alternative target variable, it does not say anything about how the user behaves, other than they are indeed a good customer.
- *culturekey* : Same as *marketname*.

The following is a list of attributes removed from the **CustomerJourney** dataset during work, along with the reason for the removal.

- *logisnw2w4* : Removed since this attribute is in the definition of our target variable *iscjretained*.
- *iscjtrial* : Removed since it is always true.
- *iscjonboarded* : Removed since it serves as an alternative target variable, and is set by us, it does not say anything about the user behaviour that is not already present.
- *iscjengaged* : Same as *iscjonboarded*.
- *iscjinvested* : Same as *iscjonboarded*.

In both datasets the *customerid* attribute is kept in each dataset, even though it cannot be used as a feature for analysis since each customer have a unique ID and thus will not yield any patterns, since it can be used to join the two datasets together.

4.3 Tree Type

While researching R, two distinct type of decision trees came up, an implementation of regular decision trees that closely follow the method described in [?], as well as conditional inference trees[4], which I will shorten to *ctree* in this report.

Both models produce binary trees that can be used to solve classification problems. Each node in the tree represents a variable and each edge out of the node contains a “case” that tells something about the variable (is it true/false, ≥ 5 , and so forth). Each leaf of the tree is then a class, there can be several leafs with the same class, they just represent different characteristics of the same class.

The main difference between a regular decision tree and a *ctree* is how it is created. A regular decision tree will choose to split using information measures such as seen in Quinlan et. al.[6, p. 89], whereas the *ctree* framework will split based on the relationship between the variables and the covariates.

Write more here. And create code for the experiment.

Write a better explanation without bloating it.(See [4].)

4.3.1 Plotting

Since a big part of the project is for Simplesite to learn about their customers behaviour on the site, as much as letting a computer learn it, the readability of the created models became a criteria as well. In order to guage how easy it was to get information about the models created by the trees I plotted models created on the same data, using different functions. Figure 2 shows the result for the different models and functions.

While which figure is easiest to read can be subjective, I have a few observations about the figures:

- 2a Displays the most information of the different figures but seems to have issues with being scaled down due to amount of information in display.
- 2b Shows most of the information of 2a without overflowing.
- 2c Only shows the edges without showing decisions.
- 2d Displays much of the same information as 2a and 2b but also seems to have problems with scaling.

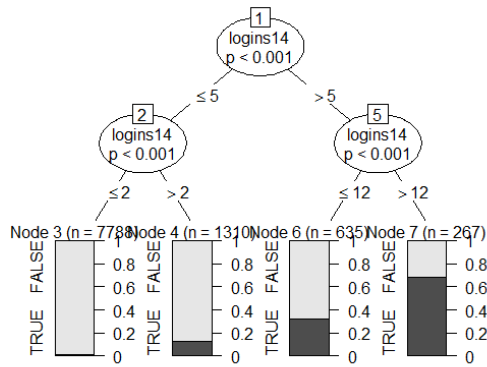
While all the examples in Figure 2 are very simple it does illustrate nicely that some of the plot functions will have trouble later when the tree grows. Based on the above examples I would use the *ctree* model with the plot function from Figure 2b.

4.3.2 Model Precision

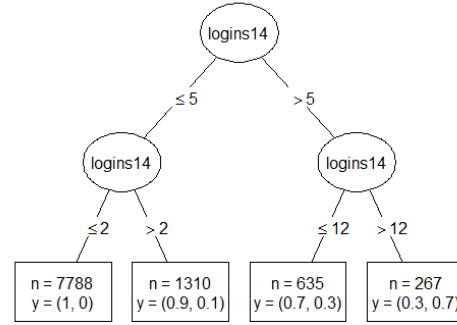
In order to make sure that we do not loose any “precision” by selecting the *ctree* model over the normal decision tree, I performed a number of tests with each of the models in order to determine if this would indeed be the case.

For the tests I performed 5-fold cross validation and different maximum tree depths and recorded the mean precision for each model. The dataset in question was the combined dataset as described in Section 4.2 (excluding the *customerid* column.) The code for this test can be seen in Appendix ??.

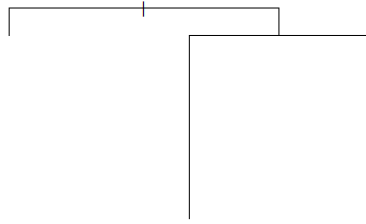
The results shown in Table 3 indicates that for our particular dataset, the accuracy difference for the two models have a very similar performance in terms of accuracy for our dataset. They both predict correctly in around 94,3% of the cases with a difference of less that 0,1% between the best and worst performer.



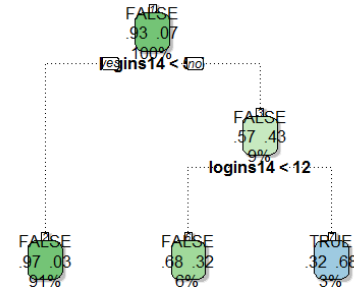
(a) The default plot of a `ctree` model.



(b) Plot of the same model using a customized plot function.



(c) The default plot of a `rpart` model.



(d) Plot of the same model using `fancyRpartPlot`

Figure 2: A side by side view of the different plots produced for the `rpart` and `ctree` models respectively. Code for creating the figures can be found in Appendix ??.

Max Depth	rpart Accuracy	ctree Accuracy
4	0.942799	0.9427990
8	0.942799	0.9431958
12	0.942799	0.9436638

Table 3: The mean accuracy for the different 5-fold cross validation runs.

Formula	Max Depth	Mean Accuracy
iscjretained .	4	0.9427990
	6	0.9429672
	8	0.9431958
iscjretained edits14	4	0.9350055
	6	0.9350227
	8	0.9350119
iscjretained logins14	4	0.9427990
	6	0.9427990
	8	0.9427990
iscjretained edits14 + logins14	4	0.9427990
	6	0.9428465
	8	0.9429414

Table 4: Mean accuracy of different formulas and tree depths using 5-fold cross validation.

An interesting observation from the data above is that the **rpart** model have the same accuracy for all three runs indicating that the model created does not change even when it is allowed to grow more complex. After plotting models from each step, I discovered that it did not grow beyond a depth of 2 and based itself solely on the **logins14** variable, similarly to what could be seen in Figure 2d.

4.3.3 Choosing a Model

Based on the two criteria highlighted in in Sections 4.3.1 and 4.3.2, I selected to go with the **ctree** package, because it was easier for me to read the information the model was created on without sacrificing any precision compared to the traditional decision tree implemented in **rapart**.

4.4 Model Analysis

For this section the two datasets **EngagementData** & **CustomerJourney** have been joined on the *customerid* attribute, into one dataset after which the *customerid* attribute was removed. This is the dataset used for the remainder of this section unless otherwise stated.

4.4.1 Dataset Features

In section 4.2 I discarded a number of features in the dataset and reasoned about why they would not be usable for our purpose. This leaves 14 features as well as the target variable, **iscjretained**. While all of these features might be usable

4.4.2 Formula and Tree Depth

5 Results

6 Competencies and Methods

7 Conclusion

References

- [1] K. J. Bekkelund. Succeeding with freemium. *Innovation and Entrepreneurship, Specialization Project*, 2011.

- [2] L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and regression trees. *The Wadsworth Statistics/Probability Series*, 1984.
- [3] cvr.dk. Enhedsdata for simplesite aps, 2016. [<https://datacvr.virk.dk/data/visenhed?enhedstype=virksomhed&id=10079861>; accessed 11-January-2016].
- [4] T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical statistics*, 15(3):651–674, 2006.
- [5] N. Pujol. Freemium: attributes of an emerging business model. *Available at SSRN 1718663*, 2010.
- [6] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.