Predicting Patient Churn: Features that Predict when Breast Cancer Patients Leave their Online Community

Anne Merel Sternheim

5774233

Utrecht University

July 9, 2018

Supervisors:
Prof. Dr. Arno Siebes *Utrecht University*
Dr. Stephan Raaijmakers *TNO*

Abstract

Often, patient organisations maintain an online community aimed at the patients they associate with. The Dutch breast cancer organisation (BVN) for example hosts the forum 'de Amazones', which is aimed at breast cancer patients. It is an important medium for patients and the association alike. for the association, it is a means to enable patients the best quality of life. For patients, participation on the forum empowers them. It was observed in the past year, however, that the activity on 'de Amazones' strongly decreased.

This thesis applies the principle of customer churn (unsubscription) to the forum as an effort to identify those forum users that will leave the forum within one, two or three months. Identifying churners is a first step towards a program for social communities like the Duch breast cancer association to identify users as churners, and respond accordingly.

The problem of churn prediction was approached from a supervised machine learning point. Twelve simple and easy-to-annotate variables were used to identify all forum posts with. Half of them described a single point in time (static features), while the other half summarised the past month (retrospective features). They were grouped into different feature groups, called: inactivity, textual, textual (retrospective), opinion, and opinion (retrospective) features. Different combinations of these variables were used to predict whether or not the writer of the post would be churned within one, two, or three months. The algorithm that was used is called XGBoost. It builds an ensemble of trees with gradient boosting. The resulting models were compared, to determine which groups of features were the most influential ones. Predictive accuracy was measured in ROC-AUC.

The results show that on a realistic test set, churn in one month can be most accurately predicted (AUC = 0.670). This result is further examined with the false negative rate (FNR), which reflects how many of churners were correctly identified. Both scores were visibly influenced whenever only few samples from the data were available. It was also shown that the two restrospective feature groups were the most influential feature groups.

Preface

The thesis in front of you was written for my graduation at Utrecht University. It was conducted at TNO, an organisation doing innovative research in a variety of domains. This thesis was conducted under the hood of TNO's 'Hematon' project. The aim of the Hematon project is to provide a user-friendly platform for both patients and professionals, with which relationships between different biomedical entities can be established from the content of patient fora. For my research, I was able to use the forum data that BVN, the Dutch breast cancer association, provided for the Hematon project.

I would therefore like to express some huge thanks to BVN. Without the data they allowed me to use, I would not have been able to write this thesis. Thank you!

Things did not always go according to plan, but I was very lucky to have Arno Siebes as my supervisor at Utrecht University, and Stephan Raaijmakers as my daily supervisor at TNO. Thank you, Arno, for the regular feedback sessions and necessary critical notes on my 'plan C'. Sorry that this is not a short thesis. And thank you, Stephan, for being so endlessly supportive![1] Thanks to you two, I have learnt an awful lot in the past nine months. I have made many mistakes, and more importantly: I have learnt from them. And I think you were right, Stephan, that someday I will look back on this period, and realise: 'I was so stupid...'

[1] Even after I sneaked up on you, twice...

Contents

Chapter 1

Introduction

Patient organisations like the Netherlands' Breast Cancer Association (BVN; Borstkankervereniging Nederland)[1] take up an important role in the communication between patients and professionals. Their overall goal is to enable patients to have the best quality of life, and ensure they can get the best medical care. BVN for example communicates the patients' perspectives on hospital treatments to medical professionals, in order to improve the medical care. To improve the patients' quality of life, they provide the community with the newest relevant information gained by researchers, and they arrange contact between patients who have had similar experiences.

One medium BVN uses to arrange contact between patients, is their online forum, called 'de Amazones'. Patients use the forum (anonymously) for a wide range of purposes, for example gathering information or staying in contact with social peers. Batenburg (2015) concluded that "being online together" increased psychological well-being of cancer patients, regardless of their offline situations. Moreover, according to van Uden-Kraan et al. (2008), patients that only read the posts of others and actively posting patients get equally empowered from their interactions with the forum. The two most experienced outcomes of forum participation, according to van Uden-Kraan et al. (2009), are feeling better informed, and enhanced social well-being. This is strongly related to the concept of empowerment. From (Lauzier et al., 2014, p. 3220):

> "Empowerment in relation to health refers to the individual's feelings of being able to manage the challenges of the cancer experience and of having a sense of control over one's life."

The forum is thus a way for BVN to enable patients to have the best quality of life. Moreover, it is of tremendous value to its active, as well as passive forum participants. This includes anyone looking up information about breast cancer on the internet, because the forum conversations can be freely accessed by anyone.
Even for researchers and other medical professionals, the forum is of added value, because there is a lot of information going around on it. For example about the side

[1] https://borstkanker.nl/

effects of medicine and how patients manage them. However, crawling the forum for information manually takes a lot of time. In order for patients and professionals alike not to have to read through all forum posts, van Oortmerssen et al. (2017) created a tool with which textual co-occurrences of biomedical entities can be investigated. This advanced search engine visualises how biomedical entities in forum posts are related to each other. This system has already lead to new insights, such as that splitting the dose of a particular medicine used by patients with gastrointestinal stromal tumors (GIST) reduces the strength of the medicine's side effects (van Oortmerssen, 2016, *Dutch*).
In short: A forum is a very valuable medium for patient organisations, patients, researchers and medical professionals alike. This is why it is important to maintain.

However, due to a confluence of circumstances, such as the reorganisation of the Dutch breast cancer association, and the increasing popularity of Facebook[2]-groups similar to the BVN-forum, BVN has noticed a strong decrease in the number of active forum users over the last year[3]. An indication of which forum users are likely to leave shortly (and ideally *why*), would help the association to formulate strategies aimed at retaining their current forum users.

As a first effort to provide associations like BVN with these insights, this thesis aims to train a model that is able to predict whether a patient will churn, based on their available forum posts. Patient churn was predicted one, two and three months into the future, using variables that were extracted from the user's forum texts. The two main questions that this thesis aims to answer, are the following:

1. How accurately can churn be predicted from a given point in time, one, two, and three months into the future?

2. Which features, measurable from a user's forum posts, are most important for the prediction of churn?

In addition, it is in the nature of the data that the predicted variable is unbalanced (Section 4.2 will handle this in more depth): It is more likely that a patient will *not* churn within three months, than that they will. This raised an additional research question:

3. How can the problem of class imbalance best be handled?

In order to answer these questions, a supervised machine learning algorithm (XGBoost) was trained on twelve features, measured from a user's forum posts. Half of these features described the forum posts within a one-hour time bin, the other half described the average forum post over the last month. The algorithm was adjusted in different ways to handle the problem of class imbalance, to predict churn in one, two and three months, and to determine which features were most important in the prediction of churn.

[2] www.facebook.com

[3] Information from personal conversations with board members from BVN, June 4, 2018.

The results show that churn can not be accurately predicted from just one hour of observation, but that a user's average posting behaviour over the last month can. Especially the 'textual features' were shown to have a good predictive accuracy. These conclusions were drawn when the trained model was tested on a balanced test set, and hold up very well when it is trained on a realistic, unbalanced test set.
All algorithms were written in the Python programming language (van Rossum, 1995). Written code is available on GitHub[4].

The thesis is organised as follows. Chapter 2 presents related work on the topic of churn. Chapter 3 provides an overview of the forum posts that were used, and how the twelve different features were extracted out of it. Chapter 4 describes the experiments that were conducted with XGBoost, and explores how unbalanced data can be dealt with. Chapter 5 presents the results. Chapter 6 discusses the findings of the experiments, and Chapter 7 finally rounds up this thesis by presenting the conclusions that were drawn from the experiments.

[4] https://github.com/annemerelsternheim/forum-churn-prediction

Chapter 2

Related work

Customer churn is when a customer ends a relationship with a company. This behaviour is most obvious when subscriptions are considered. Consider for example a customer with a telephone subscription. The moment this customer decides to quit their subscription (for example because another provider has a more attractive offer) is the moment the customer churns. It is a concept of interest for the telephone provider, because of several reasons.

For example, the customer churn rate is used to indicate how 'well' a company is. The customer churn rate reflects the number of customers that quit the subscription in a predetermined time frame, compared to the total number of customers that was subscribed to this service at the beginning of that time frame. A low churn rate is assumed to reflect customer satisfaction, and indicates that a company is doing well. A high churn rate on the other hand, may indicate that the company is not providing the appropriate customer care (Rust and Zahorik, 1993; Ahn et al., 2006), or that other companies are making more attractive offers.
Another reason why churn is of interest to companies, is that retaining customers is often less cost-expensive than the acquisition of new ones (Gallo, 2014). When a company is able to cut costs on customer acquisition, it will be more profitable.

There can be a multitude of reasons for a customer to end their relationship with a company. As mentioned before, a customer may have found a better offer elsewhere, or they may be dissatisfied with the company's serivices and customer care. Many companies invest in a strong *customer-brand relationship*, in order to retain their customers. According to Casidy et al. (2018), studies have shown that there is indeed a (positive) correlation between the strength of the customer-brand relationship and the likeliness that a customer will continue their relationship with the company. The customer's involvement with the company also influences the relation. In addition, Verhoef (2003) found that programs aimed at improving the customer-brand relationship are able to reduce churn rates.

The principle of customer customer churn can be applied to any kind of business, including online social communities like 'de Amazones'. Although such platforms do not have customers in the traditional sense, it can be informative to analyse the activity pattern of its users. This has been done frequently already, but there are two recurrent issues that are the focus of many of the papers published within this research area: how do we define churn, and which features can we use to predict it?

The first issue is the *definition* of churn. When a customer quits a subscription to a service with a company, this is a clear and unambiguous moment of churn. On social platforms, however, this moment is only rarely as clear. Karnstedt et al. (2010) proposes to determine churn on social platforms as a user's relative inactivity, as compared to an earlier time frame. This definition is based on the idea that churn is closely related to 'user engagement': In order to determine how engaged a user is with the web site, measures such as frequency of visiting, clicks and total time spent on a web page are used. In their paper, Lehmann et al. (2012) use three different kinds of engagement metrics to identify models of user engagement: popularity, activity and loyalty. Most closely related to Karnstedt et al.'s definition of churn, is loyalty. It reflects how often users return to the same website, and how much time they spend on it. The benefit of Karnstedt et al.'s definition of churn is that it can be applied on any platform and in any context, as long as one can determine a user's past and present level of activity. However, the definition flags both permanent (comparable to quitting a subscription) and temporary churn (comparable to skipping a few sports classes due to a holiday). Therefore, in order to flag only permanent churn, for this thesis an individual's churn was determined as the last log-on date that was registered for this user.

The second issue is the choice of *features*. In subscription-providing companies, there is often a lot of information readily available about a customer, such as their other subscriptions and purchases with the company, social background, economical background, conversations with the customer service desk, and so on. For online social platforms, such information is usually unavailable. This may have to do with the fact that one does not usually provide all this information, just to participate in an online community.
A lot of research, including this one, is aimed at determining which features are suitable to determine churn in social communities. Some researchers suggest (e.g., Karnstedt et al., 2010) that external events should be taken into account, such as when similar platforms become available, or when a relevant topic loses or gains interest. Others suggest that the (social) network between users should be considered (Kawale et al., 2009; Karnstedt et al., 2010, 2011; Richter et al., 2010). The idea behind this is that connections in an online social network influence each other: when someone churns, this may cause their connections (friends) to start churning as well.Again other researchers propose that an individual's level of engagement or activity should be considered (Kawale et al., 2009; Richter et al., 2010; Lehmann et al., 2012). This type of feature mainly focusses on how often and for how long a user is online. Similar is the approach that uses participation variables to determine churn (Yang et al., 2010): with this approach,

the participation of a user is classified with features such as the number of questions answered and asked, or the number of up- or downvotes for each of their comments. Because the last two approaches consider only the user, and not their community or external features, they are most similar to how churn is determined for subscription-providing companies. Moreover, research on churn in online communities often proposes a combination of two or more of these types of features, as they will often complement each other.
For this thesis, the question was whether and how *post-based features* could be used to determine churn on fora. As a consequence, only those features were annotated that could be measured from a single user's forum posts. This approach is most similar to that of Yang et al., who used a single user's participation pattern to determine when a user would churn.

These decisions about the definition of churn and the choice of features are important in this thesis to answer the research questions that were posed in the introduction. The next chapter, Chapter 3 will describe the choice of features and the way they were determined. In Chapter 4 the experiments are described. They were constructed such that they would answer the research questions **how accurately can churn be predicted**, and **which features are most important**? In addition, the **imbalance** of the data set was explored.

Chapter 3

Data

The forum that was used to extract features from, is called 'de Amazones'. It is under the wing of the Dutch breast cancer association BVN ('borstkankervereniging Nederland'), a partner of TNO's Hematon-project. The forum is aimed at those diagnosed with breast cancer, but also friends and relatives of breast cancer patients are known to join the forum.

The content of conversations on the forum can be accessed by members and non-members alike. However, starting a conversation or responding to one is restricted to members only. Members of the forum have a personal profile, which they can fill in to any degree they see fit. The only obligation they have is to choose a user name, which will be visible when they participate in a conversation, and on their profile.
Members can start conversations in different ways. They can write a personal story, they can write someone else a private message, or they can start a forum thread. In this thesis, only the latter is used to collect data from. The topic of forum conversations can be anything: from forum-related topics like how to post a picture, to cancer-related topics like the side-effects of certain medicine.

A forum thread has two main components: the *thread start*, and the *responses* to it. Typically, a thread is started because a forum member faces a problem, and would like to know from other members who have faced similar problems, how they have dealt with it. In this thesis, when the difference between the two is unimportant, both thread starts and responses will be called *(forum) posts*.

3.1 Forum users

At the time of data collection (April 18, 2018), 5630 users were registered with the forum. Only 2266 (40%) of these users had posted at least one forum post in the past thirteen years (the first post in the data set is from September 1, 2004)). The absence of most of the remaining 3364 users on the forum may be explained by the type of activity the forum users prefer. As mentioned before, the platform not only allows communi-

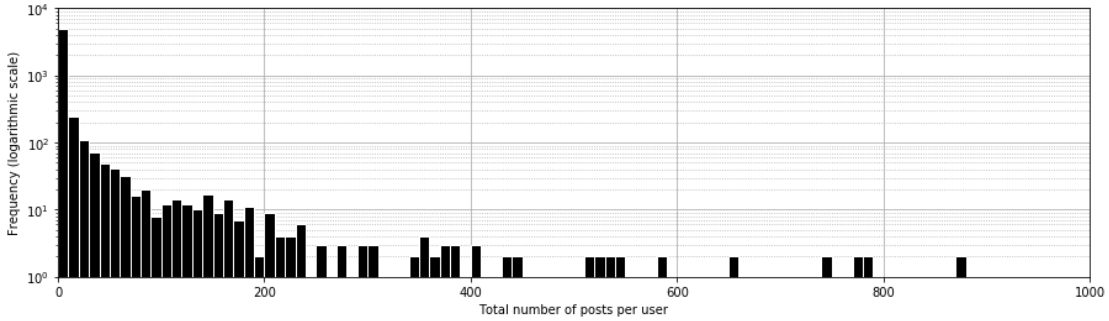


Figure 3.1: Histogram showing the distribution of number of posts per forum user. A logarithmic scale is used for frequency (y-axis), to visualise the distribution more clearly.

cation though forum threads, but also through personal stories and private messages. Moreover, a special kind of user is the *lurker*: they only watch the activity of others, and never show any activity themselves (van Uden-Kraan et al., 2007, 2008).

Figure 3.1 visualises the total number of posts each user posted. On the x-axis is the total number of posts of a user, with bin size = 10. The y-axis shows how frequently this total occurred in the data. The y-axis was scaled logarithmically. From the figure it becomes obvious that of all $n = 5630$ forum users, most ($n = 4881$) posted 10 posts, or less. The remaining users ($n = 785$) posted 11 or more posts. The total number of posts of 16 users is not included in this figure. They posted between 1017 and 4253 posts. These were not shown in the histogram, to reduce cluttering of bins on the lower hand of the x-axis. The total number of posts available in the data is 122697.

3.2 Data collection

BVN's business partner YouWe[1] collected all BVN forum posts that were used for this thesis. Each post that was posted after September 1^{st} 2014, was stored in a database (.json), and was identified with meta-data such as the post date, the author ID, and its HTML-formatted content. Listing 3.1 shows how one post entry from the database is built up. The content of the post (the 'body') is formatted in HTML. Entries in a related database (.json) contained meta-data about the different users that were subscribed with the forum, such as their author ID, first day of forum access, and last day of forum access. Listing 3.2 shows how one user entry from the database is built up.

3.2.1 Preprocessing of the data

For this thesis, the HTML-formatting of the content of the forum posts was irrelevant. Therefore, all formatting was stripped off such that only plain text would remain. An

[1] www.youwe.nl

Listing 3.1: An example post entry from the database

```
{
 'body': '<p>Hi,</p>\n<p>I have a question. <br/>\nDoes anyone
    know the <i>answer</i>? I would be so grateful! <img alt="
    smiley" height="23" src="https://.../regular_smile.png"
    title="smiley" width="23" /> </p>\n',
 'forum_id': '222',
 'forum_topic_id': '24,222',
 'post_date': '22/02/2017 - 02:22',
 'title': 'The post content is in HTML-format',
 'updated_date': '22/02/2017 - 02:22',
 'user_id': '2222'
}
```

Listing 3.2: An example user entry from the database

```
{
 'avatar': 'http://.../image.jpg',
 'created_date': '02/02/2002 - 02:02',
 'last_access': '02/12/2017 - 20:20',
 'last_login': '02/12/2017 - 22:22',
 'name': 'MyUserName22',
 'roles': 'De Amazones User',
 'status': 'Actief',
 'user_id': '2222'
}
```

Listing 3.3: The content of a post, plain text only.

```
'Hi, I have a question. Does anyone know the answer? I would be
    so grateful! (EMO:smiley)'
```

Table 3.1: Different occurrences of a smiley face in the data, and its corresponding tag.

	forum	HTML	tag
ASCII	:) / :-)	:) / :-)	(EMO:SMILEY)
Unicode	☺	☺	(EMO:SMILEY)
From drop-down menu		*see Listing 3.1*	(EMO:SMILEY)
Double colons		:wink:	(EMO:WINK)
Square brackets		[wink]	(EMO:WINK)

example is given in Listing 3.3. The example also shows how one occurrence of an emoticon (smiley) was substituted with a tag `(EMO:smiley)` in order to further improve readability, and to ensure that characters from the image link were not misinterpreted by for example the sentence tokeniser as indicating the end of a sentence.
Links to external websites, as well as images other than emoticons, were treated in a similar way, for the same reason. They, too, were substituted with a tag: `(LINK)` or `(IMG)`, respectively. Quotes from other forum users were removed altogether, as they were assumed to not add any information about the author of the post.

There are different ways to 'express' emoticons, and several were used on the forum. Probably the easiest to interpret is the emotion icon: an image of a gender-less, often yellow face, expressing a specific emotion or mood. This kind of emoticon could be inserted in forum texts in different ways. Users could click the desired emoticon in a drop-down menu, but they could also type the name of the emoticon between double colons or square brackets. Table 3.1 gives examples of this kind of emoticon. The table shows the different ways the forum user could insert the emoticon, how it looked in the HTML-formatted posts provided by youwe, and which tag was used to substitute the emoticon.
Two other kinds of emoticons were used on the forum. The most common one is the sideways ASCII emoticon. This kind of emoticon consists only of characters from the ASCII alphabet and punctuation set. the sequence ':-)' for example indicates happiness, and ';-)' a wink. The (semi-)colon represents the eyes of the figure, and the bracket the mouth. The hyphen represents a nose, and is often omitted.
The last kind of emoticon is the unicode emoticon, and is uncommon. In all forum posts since September 1^{st}, 2014, only two occurred: one heart and one smileyface. Both the ASCII and unicode smiling emoticons are shown in Table 3.1, together with their HTML representation, and the tag they were substituted with.

In addition, all non-ASCII characters were replaced by their nearest ASCII neighbours, using the Python package `Unidecode` (Solc, 2018). For example, the Dutch 'patiënt' (a *patient)* contains the non-ASCII character 'ë', which was substituted with its nearest ASCII neighbour 'e'. As Python's default encoding is ASCII, reading or writing such a string without specifying a different encoding would cause a `UnicodeEncodeError`. Removing all non-ASCII characters once was a very convenient way to avoid `Unicode-`

`EncodeErrors` during all future data handling, while at the same time the posts were made and kept human-readable.

3.2.2 Feature selection

A selection of twelve independent variables has been made, which were measured from a user's forum posts. Five of these describe a user's forum posts in terms of the author's expressed opinion, and textual properties such as the length of sentences. A sixth feature reflects the inactive time between a user's two successive posts. Together, these six features are referred to as the 'static' features, because they reflect a post at a given point in time. The remaining six features are retrospective adaptations of the static features: they reflect the average value of a feature over the past 30 days.

In order to determine whether churn can be more accurately predicted one, two or three months into the future, three binary dependent variables were annotated in the data: churn in 1 month, churn in 2 months and churn in 3 months. An overview of all twelve independent, and three dependent variables is given in Table 3.2.

In order to explore the individual effect of the independent variables on the prediction of the dependent variables, different combinations are used to build a predictive model with machine learning. Section 3.2.4 will go deeper into how the features were combined.

Static independent variables

All variables in this section describe one post from one author, at one point in time. An overview of all six static variables is given in Table 3.2b. The variables were chosen for their expected predictive ability, together with their ease to annotate.

Question_marks: Breast cancer patients that join 'de Amazones' are typically looking for contact with social peers, or for information. A customary way to get information, in both the offline and online world (including fora), is to ask questions. It seems reasonable that when a user came to the forum to ask questions, they will stop participating actively when they are out of questions. The number of questions asked, may therefore be a good predictor of churn.

Sentiment and *Subjectivity:* Opinion mining is a term that refers to both the analysis of sentiment, and the analysis of subjectivity of a text (Bo Pang, 2008). It is believed that the sentiment and subjectivity values that are determined with opinion mining, reflect the author's emotions and personal involvement with a subject.
On the forum, many different subjects of conversation are considered. Although it is beyond the scope of this thesis to determine what subjects the forum users are writing about, their opinions will be mined. According to Barone et al. (2000), participants judged brands and their extensions more positively when they were in a more positive mood, and more negatively when they were in a negative mood. If a patient's mood

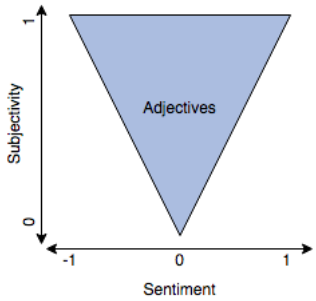


Figure 3.2: An abstract representation of the triangular relationship between sentiment and subjectivity of adjectives.

has influence (either negatively or positively) on their opinion about all kinds of subjects, this can be picked up by the opinion mining techniques. It will be interesting to see whether a patient's general mood has influence on when they decide to churn, and whether it is a good predictor of churn.

De Smedt and Daelemans (2012b) present an open-source lexicon for Dutch adjectives, which is part of the Python Package `Pattern` (De Smedt and Daelemans, 2012a). All adjectives are annotated with a subjectivity value of 0 to 1 (0 denoting complete objectivity), and a sentiment value of -1 to 1 (the distance from 0 denoting the strength, and the polarity (+ or -) denoting the polarity of the sentiment (positive or negative)). Sentiment and subjectivity are related in a triangular way. This means that adjectives with a high (positive or negative) sentiment value are also more subjective, but subjective adjectives can also be of neutral sentiment. Figure 3.2 illustrates this idea.

Post_length and *Sentence_length:* Yang et al. (2010) observe that those who ask longer questions, and those that provide longer answers on Q&A platforms, tend to use the platform for a longer period of time. Healey and Hoek (2014) also observed that the length of posts was a good indicator of churn in a smoking-cessation community. It is therefore expected that the same will hold for 'de Amazones'. The length of posts was measured in sentences, and the length of sentences was measured in words.

Inactivity: The sixth feature is different from the others. While the other five features looked at one post only, the Inactivity feature compares two in order to determine how much time had passed between the two successive posts. The need for this variable was motivated by the wish to capture how active a forum member is. A similar measure was used by Karnstedt et al. (2010) in their definition of churn. They defined churn as the moment that a user's average activity over a previous window has dropped to less than a predetermined fraction of their average activity in the previous time steps. This definition leans on the previous activity (PA) of a forum user.

Retrospective independent variables

The six features motivated in the prior section, Question_marks, Sentiment, Subjectivity, Post_length, Sentence_length and Inactivity, all have a retrospective counterpart. This means that instead of describing one single post like the static features did, the retrospective features describe the average (Sentiment / Post_length / ...) value of all a user's posts that were posted in the last 30 days.

Table 3.2c shows an overview of all six retrospective variables. As these retrospective features reflect a larger portion of a user's data leading up to the moment of possible churn, they are expected to be better predictors of churn than their static counterparts.

Dependent variables

Three dependent binary variables were annotated in the data, reflecting a user's churn in one, two or three months in the future. They are shown in Table 3.2a. When customer churn is considered, the moment of churn is usually defined as the moment the customer ends their subscription with the company. However, only a minor portion of forum users will eventually unsubscribe. In the data used for this thesis for example, of all 5630 forum users, 4501 had not been active on the forum for at least one year (79.95%), while only 269 users' accounts were actually inactive at the time of data collection (4.78%). As mentioned before in Chapter 2, Karnstedt et al. (2010) formulated a definition of churn which relies on a user's relative level of activity. This definition is therefore independent of the date of unsubscription, and ideal for online platforms. However, the definition is unable to differentiate between temporary and permanent churn. As in this thesis only temporary churn is of interest, the moment of churn for each user was defined as the last date on which a user had logged onto the forum.
For every hour between the user's first post on the forum and their churn, it was determined whether the user would still be active in one, two, or three months. The resulting labels were used to train a churn-predicting model, and later to test its predictive accuracy.

3.2.3 Feature annotation

The six static features described in Section 3.2.2, were determined for every post. These posts were then aggregated into hour-long time bins. The retrospective values of all static independent variables, as well as the values of the three dependent variables, were determined for every time bin.
Figure 3.3 shows these time bins as rectangles, attached to a forum user's time line. This particular (fictional) user posted their second and third post within the same hour, which therefore ended up in the same time bin. For the fifth post, the figure shows how far back a retrospective variable looks. Finally, the figure shows that those time bins that occurred within three months before the user's churn date were labeled 'True', for 'churn in three months' (shown as green).

Table 3.2: An overview of all annotated variables

(a) The dependent variables

Name	Description
Churn in one month	*Boolean.* Whether a patient will churn (1) or not (0) within the next 30 days
Churn in two months	*Boolean.* Whether a patient will churn (1) or not (0) within the next 60 days
Churn in three months	*Boolean.* Whether a patient will churn (1) or not (0) within the next 90 days

(b) The independent variables (static)

Name	Description
Question_marks	*Float.* The number of sentences ending in at least one question mark, divided by the total number of sentences.
Sentiment	*Float.* A textual reflection of the emotion of the author: on a continuous scale of very negative (-1) to very positive (-1).
Subjectivity	*Float.* A reflection of the personal content in the author's text: on a continuous scale from very objective (0) to very subjective (1).
Sentence_length	*Float.* The average length of the sentences occurring in the text, in words.
Post_length	*Float.* The length of a text, in sentences.
Inactivity	*Integer.* The time that passed between two of a user's successive posts, in hours.

(c) The independent variables (retrospective)

Name	Description
Question_marks_retro	*Float.* The number of sentences ending in at least one question mark, divided by the total number of sentences, as an average over the past 30 days.
Sentiment_retro	*Float.* A textual reflection of the emotion of the author: on a continuous scale of very negative (-1) to very positive (-1), as an average over the past 30 days.
Subjectivity_retro	*Float.* A reflection of the personal content in the author's text: on a continuous scale from very objective (0) to very subjective (1), as an average over the past 30 days.
Sentence_length_retro	*Float.* The average length of the sentences occurring in the text, in words, as an average over the past 30 days
Post_length_retro	*Float.* The length of a text, in sentences, as an average over the past 30 days.
Inactivity_retro	*Float.* The time that passed between two of a user's successive posts, in hours, as an average over the past 30 days.

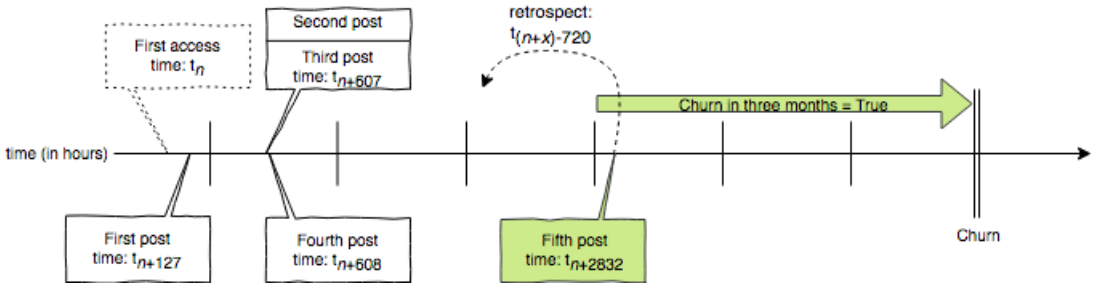


Figure 3.3: The time line of one user. At the far right: the moment of churn. At the far left: the moment the user first logged onto the forum. The boxes in between reflect time bins (of an hour). Time bins which were labeled 'true' with respect to the dependent variable 'Churn in three months' are colored green. Finally, the arrow pointing leftwards shows the retrospective time frame of 30 days (720 hours), relative to one of the time bins. Empty time bins are not shown in this figure.

The current section describes the methods that were used to determine the values of all annotated variables. After annotation, the features were scaled using the `MinMaxScaler()` from Sklearn's (Pedregosa et al., 2011) preprocessing module. This scaler transformed every feature to a given range from 0 to 1, thereby preserving the 0-values occurring in the data. Although normalisation of the data is not necessary for tree boosting, it makes it easier to compare the results of this thesis with future results obtained with other models.

Static independent variables

Question marks: In order to determine the number of questions that was asked in a forum post, the number of sentences ending in at least one question mark was determined. A selection of the results was checked manually and confirmed the validity of this approach. First, sentences were split from each other using NLTK's (the Natural Language Toolkit, Loper and Bird, 2002) sentence tokeniser. Second, a regular expression was used to determine whether a sentence contained at least one question mark. The expression was formulated as follows:

```
\?+
```

In which the \ is used to escape the 'special' meaning of the `?`, and the `+` is used to match any sequence of at least one `\?`. If a sentence matched the regular expression, it was recognised and counted as a question. Explicitly stating the end of a sentence was unnecessary, because the sentence tokeniser splits sentences based on interpunction: all (groups of) signs are assumed to mean that a sentence has ended, except some which are not followed by a white space. This means that if a question mark were placed (unconventionally) in the middle of a sentence, the sentence tokeniser would split the

sentence directly after the question mark, and the first part would be recognised as a question. A manual check of the results gave no reason to consider an alternative approach, which would bypass this possible scenario. In addition, using this approach, sentences that ended in multiple signs, amongst which were at least one question mark, were also recognised as questions (See example below).

The following are some sentences which would have matched the regular expression, had they occurred in the data. Matching sequences are underlined:

- 'Hello?'
- 'Are you serious??!!'
- 'And the answer is....????'

The number of question marks was then divided by the total number of sentences, resulting in a fraction that expressed how prominent the question was in the post. For example: if one of two sentences was a question, the post was annotated (1/2 =) 0.5. Figure 3.4a shows how all annotated and normalised values of the questions feature were distributed. The bins in this figure increase logarithmically in size, which makes the lower frequencies occurring at the higher percentiles more visible. In addition, the y-axis of the figure is discontinuous, allowing the lower part of the figure to zoom in on the lower frequencies, while at the same time showing up to where the highest frequency reaches. The highest frequency occurs at the lowest percentile. This was expected, as bins between posts are empty, and outnumber the bins that do contain posts (filled:empty = 54894:546230). As empty bins contain no post, they contain no text and hence no question marks. In the upper right corner of the figure, the minimum and maximum values of the non-normalised data are shown. These allow us to conclude from the figure that in most posts, about one in five sentences is a question.

Sentiment and *Subjectivity:* Both Sentiment and Subjectivity values were determined with Pattern (De Smedt and Daelemans, 2012a), a toolkit for natural language analysis. In order to rate a text in terms of sentiment and subjectivity, the package uses a lexicon of Dutch adjectives (De Smedt and Daelemans, 2012b, an english lexicon is also available). The accuracy of this opinion analysis is about 82% for book reviews[2]. The function takes as input a text of arbitrary length, and outputs the sentiment and subjectivity value of the text, based on De Smedt and Daelemans' adjectives lexicon. The function is able to deal with different meanings based on word order, negation and interpunction. Sentiment values range from -1 up to and including +1, where negative values denote negative sentiment, and positive values denote positive sentiment. Subjectivity values range from 0 to 1, where 0 denotes complete objectivity, and 1 complete subjectivity.

[2]From CLiPS's (Computational Linguistics and Psycholinguistics Research Center, Antwerpen University) online resources: https://www.clips.uantwerpen.be/pages/pattern-nl

Table 3.3: Sentiment and Subjectivity of several example sentences, as determined with Pattern

	Sentence (*translation*)	Sentiment	Subjectivity
1	"Mooi" (*nice/pretty/beautiful*)	0.70	1.00
2	"Je hebt een mooi T-shirt aan." (*You are wearing a nice T-shirt*)	0.07	1.00
3	"Wat zie je er mooi uit!" (*You look beautiful!*)	0.88	1.00
4	"Niet slecht!" (*Not bad!*)	0.44	0.90
5	"Oh nee :(" (*Oh no :(*)	−0.75	1.00
6	"Dit is blijkbaar het resultaat." (*Apparently this is the result.*)	0.00	0.60
7	"Ik hou niet van taart" (*I do not like pies*)	0.00	0.00

Because the function is able to return the sentiment and subjectivity values of a text of arbitrary length, there was no need to split the post into sentences. Instead, the entire post was fed to Pattern's opinion mining tool, which then returned its sentiment and subjectivity values. Table 3.3 shows some example sentences, together with their sentiment and subjectivity values. The parts of the text contributing to the scores (mostly adjectives) are underlined.

The first example in the table shows that 'mooi' is a very subjective word (1.00), with positive sentiment (0.70). The second sentence shows that only adjectives and adverbs (in this case 'mooi', again) are used to determine the sentiment and subjectivity of a sentence. As there are no other adjectives present in the sentence, the sentiment and subjectivity values are identical to those of the first example. Sentence three shows how Pattern handles exclamation marks: again, 'mooi' is the only adjective in the sentence, but the sentiment value is higher than in Sentence 1 and 2. This is because the exclamation mark is treated as an 'intensifier': it increases the positive (or decreases the negative) sentiment with $^1/_4$ of the original sentiment value. In this case: $1^1/_4 * 0.70 = 0.875 \approx 0.88$. The fourth sentence shows that Pattern can deal with negations: while 'slecht' by itself has negative sentiment (-0.70), its negation has positive sentiment (0.35). More specifically, a negation is assumed to reverse the polarity ($+/-$) of the sentiment, and to decrease its intensity by $^1/_2$: $-0.70 * -^1/_2 = 0.35$. The exclamation mark again increases the sentiment ($1^1/_4 * 0.35 = 0.4375 \approx 0.44$). The next sentence contains no adverbs or adjectives, but it does contain a sad ASCII emoticon (See Section 3.2.1 and Table 3.1). That is what gives this subjective sentence its negative sentiment (-0.75).
As motivated by the triangular relationship between sentiment and subjectivity (See Section 3.2.2 and Figure 3.2), there also exist adjectives with neutral sentiment and high subjectivity. An example is shown in sentence 6: it contains the adjective 'blijkbaar' (*apparently*), which has neutral sentiment (0.00) and medium high subjectivity (0.60). The last sentence is an example of a sentence without any adverbs or adjectives. Although this sentence does express an opinion, it is not detected with Pattern, because

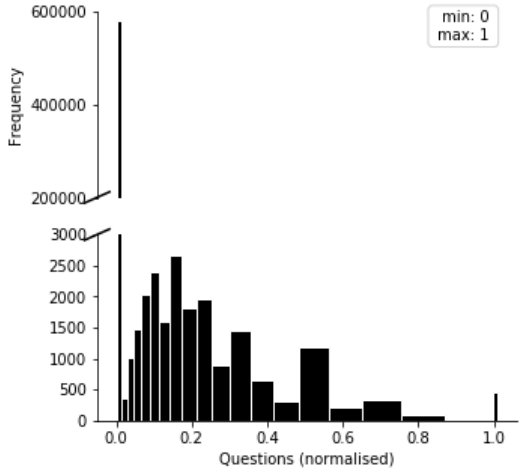
the sentence does not contain adjectives. For the scope of this thesis, this behaviour was accepted. However, for research with a more prominent role for opinion mining, it is recommended to look into ways to expand the used lexicon with nouns and verbs. In a nutshell, Pattern's `sentiment` function determines the sentiment and subjectivity of a given text, based on the occurrence of certain adjectives and adverbs in the text.

The values that were returned by Pattern's functions for opinion mining were the values that were annotated. Figure 3.4c shows how all annotated and normalised values of the sentiment feature were distributed. The bins in this figure increase logarithmically in size, starting from the middle (0.5). This makes the lower frequencies towards the upper and lower percentiles more visible. The y-axis is split, allowing the lower part of the figure to zoom in, while simultaneously showing the highest frequency in the upper half. The highest frequency occurs exactly in the middle, at 0.5. Using the minimum and maximum values of the feature, shown in the upper right corner of the figure, we can deduce that the sentiment value '0' is at the 50^{th} percentile. This explains the high frequency there, as this value occurs in all empty bins, which outnumber the bins that contain posts. The figure further shows that the sentiment of posts was generally a little more positive than negative, and that extreme sentiment values were rare.

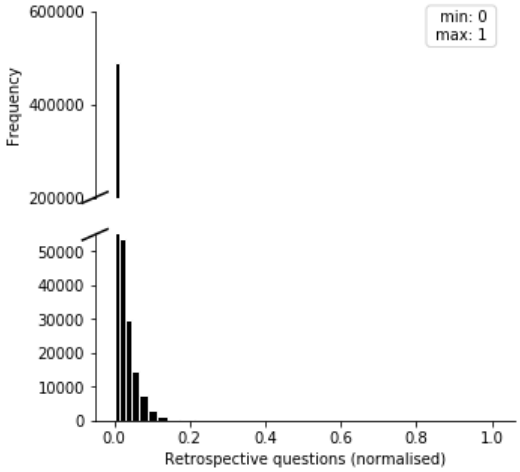
Figure 3.4e shows how all annotated and normalised values of the subjectivity feature were distributed. The bin size in this figure increases logarithmically towards higher percentiles, allowing better visibility of the lower frequencies in the higher percentiles. As before, the y-axis is split: the bottom half of the figure zooms in on the lower frequencies, while the top half shows the highest. The highest frequency corresponds to the lowest percentile, which reflects the annotated value '0': the default when a bin contains no post. Using the minimum and maximum values shown in the upper right corner of the figure, it can be deduced from the figure that the subjectivity value of most posts is about 0.4, which is slightly more objective than subjective.

Post_length: In order to determine the number of sentences in a post, the post was split into sentences using the NLTK's sentence tokeniser. Although it is possible to train a tokeniser on a specific data set, a pre-trained instance was used. This is the function `sent_tokenize`. This decision was made because the pre-trained tokeniser is generally known to perform well. A manual check of the results revealed that the tokeniser did indeed perform well generally, although it failed to recognise most Dutch abbreviations (e.g., 'o.a.' (*a.o.*)). This caused some texts to be split in unintended places, increasing the apparent length of a post in sentences.

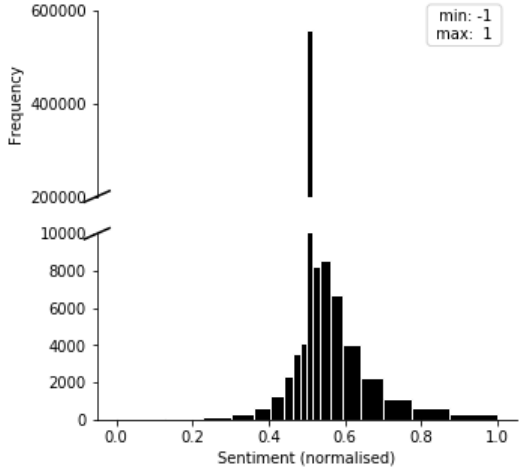
Figure 3.5a shows the distribution of the normalised post length feature. Again, the bin size in the figure increases logarithmically such that lower frequencies at higher percentiles become more visible, and the y-axis is split such that extra focus can be put on the lower frequencies. The highest frequency in this figure is at the lowest percentile, and its value is equivalent to the number of empty bins: 546230. The figure further



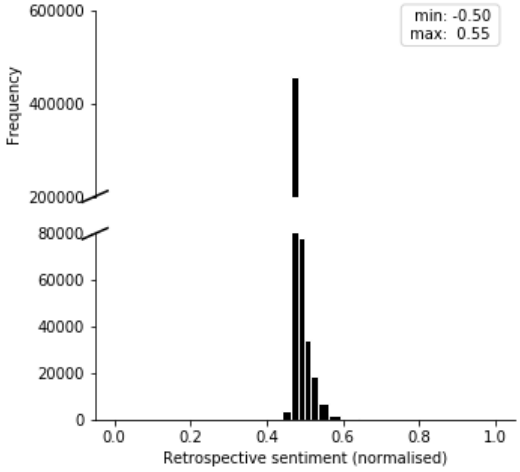
(a) Questions



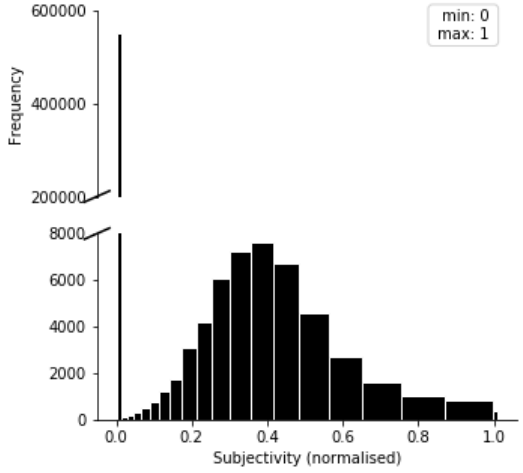
(b) Questions, retrospective



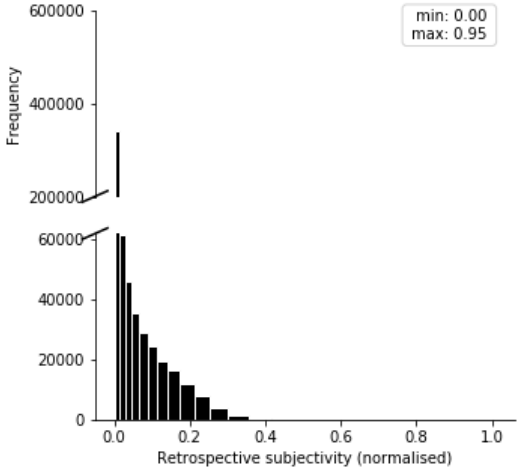
(c) Sentiment. Bins are smaller towards 0.5.



(d) Sentiment, retrospective. Bins are smaller towards 0.5.

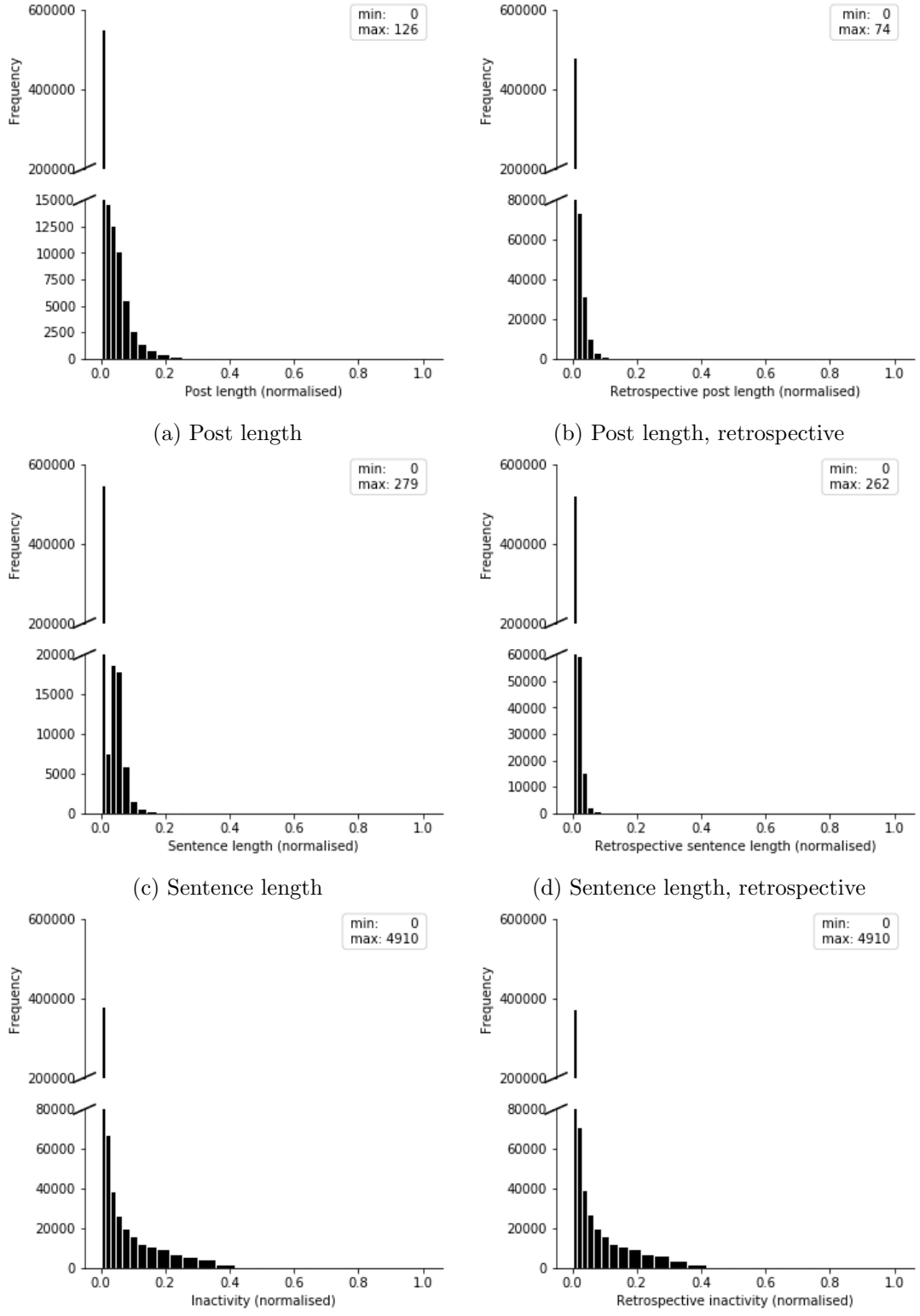


(e) Subjectivity



(f) Subjectivity, retrospective

Figure 3.4: The distribution plots of the static and retrospective features: Questions, Sentiment and Subjectivity. The y-axis is cut, such that the reach of the highest frequency would remain visible on the top half of the figure, while zooming in on the bottom half of the figure. To further increase the visibility of lower frequencies (generally higher percentiles), the bins increase in size (exception: see Sentiment). In the upper right corner of each figure, the minimum and maximum values of the non-normalised data are shown.



(a) Post length

(b) Post length, retrospective

(c) Sentence length

(d) Sentence length, retrospective

(e) Inactivity

(f) Inactivity, retrospective

Figure 3.5: The distribution plots of the static and retrospective features: Post length, Sentence length and Inactivity. The y-axis is cut, such that the reach of the highest frequency would remain visible on the top half of the figure, while zooming in on the bottom half of the figure. To further increase the visibility of lower frequencies (generally higher percentiles), the bins increase in size. In the upper right corner of each figure, the minimum and maximum values of the non-normalised data are shown.

shows that the distribution has a very long and flat tail. The longest post that occurred in the data contained 126 sentences, as shown in the upper right corner of the figure. Apparently, such a post length is very uncommon: this is reflected in the long and flat tail of the distribution. Most posts contain less than thirteen sentences: about one tenth of the maximum.

Sentence_length: The length of sentences in a post was determined as the average number of words in a sentence in a post. Again, the NLTK's sentence tokeniser was used to split a post into sentences. Then, all sentences were split into words using NLTK's word tokeniser. More specifically, NLTK's word tokeniser splits sentences into *meaningful parts*, amongst which are words, punctuation and other signs (e.g., currencies), abbreviations, and numerical sequences (e.g., 1.99). This means that the real length of sentences in words is usually less than the length determined by the word tokeniser. Because the overestimation of words in sentences is consistent, however, this behaviour was accepted. In order to determine the average sentence length of a post, the sum of all sentence lengths was divided by the complete number of sentences in the post.

Figure 3.5c shows the distribution of the normalised sentence length feature. The bin size increases logarithmically, and th y-axis is split, as was the case for similar figures presented earlier in this thesis. This distribution has a very long and flat tail: The maximum annotated value in the non-normalised data was 279 words in a sentence, but such lengths were very uncommon. Most sentences contained only 5 to 10% of this maximum: about 14 to 28 sentences.

Inactivity: In order to determine the inactivity of a user at a given moment anywhere between their first and last activity on the forum, the last time bin which contained a post was compared to a given time bin. Both time bins had time stamps in string format, and were converted to DateTime objects, using the Datetime module[3]. DateTime objects can be compared using some mathematical operators ($-$, $>$, $<$), so in order to determine the number of hours between their last post and the selected time bin, the first was subtracted from the latter:

$$\text{Inactivity} = \text{current time stamp - time stamp of last bin containing post}$$

This feature was annotated for bins that did, and bins that did not contain any posts. The inactivity value of a user's first post was determined as the number of hours between their first log on, and their first post. Figure 3.5e shows how the annotated values of this feature were distributed. The time bins increase logarithmically in size, and the y-axis is split. This allows a good look at all bars, including the ones with lower frequencies. The highest frequency occurs at the lowest percentile, which corresponds to when a time bin after a time bin which contains a post is annotated. The fact that this value is so high, may indicate that most forum users take their time for a visit to the

[3] https://docs.python.org/3/library/datetime.html

forum, or that they regularly check in on the forum. The longest time inactive between two time bins, is 4910 hours (see upper right corner of the figure). This equals almost 205 days. However, users are typically not longer inactive than 2000 hours (83 days; or 2 months and three weeks) before they post (again), or churn.

Retrospective independent variables

After the value of all features was determined for all users' time bins, the average over the previous (30 days $*$ 24 hours =) 720 hours was calculated. Figure 3.4 and 3.5 show the distributions of the six retrospective variables on the right-hand side. The distributions shown in these plots, compared to their static counterparts on their left, are typically narrower, and contain higher frequencies for the non-zero percentiles, but lower frequencies for the zero-percentile. All of this is explained by the way these variables were annotated: calculating averages decreases the influence of outliers, which typically decreases the maximum value of the feature. At the same time, this causes values to be less wide-spread (narrower distributions, higher frequencies for non-zero percentiles), and to be more nuanced (lower frequencies for zero-percentiles).

Dependent variables

In order to determine whether at a given point in time (time bin), a given user would churn within three months, their churn date was compared with the time bin. As before, the time stamps (in string format) were converted to DateTime objects using the Datetime module. Then, the bin time stamp was subtracted from the churn time stamp. A boolean function checked whether the time between these two was more, or less than three months. If the distance between the two was more than three months, the bin was annotated `'Churn in 3 months' = 0`. If the distance was less than three months, the bin was annotated `'Churn in 3 months' = 1`. The same was done for two months and one month into the future.

All three dependent variables are very imbalanced. The ratio positive:negative samples for churn in one month is 1:239, for churn in two months 1:94, and for churn in three months 1:55. Figure 3.6 shows the distribution of these three variables. It clearly shows that churn in three months contains more positive samples than churn in two months, which in turn contains more positive samples than churn in one month. Section 4.2 will discuss how the problem of class imbalance was treated.

3.2.4 Importance of features

In order to find out which features are the most accurate predictors of churn, different models have been trained, which used different combinations of features. The different feature combinations were combined with the three different dependent variables that predict churn in respectively one, two and three months into the future. Using Formula 3.1, the total number of possible feature-combinations can be calculated. n is the

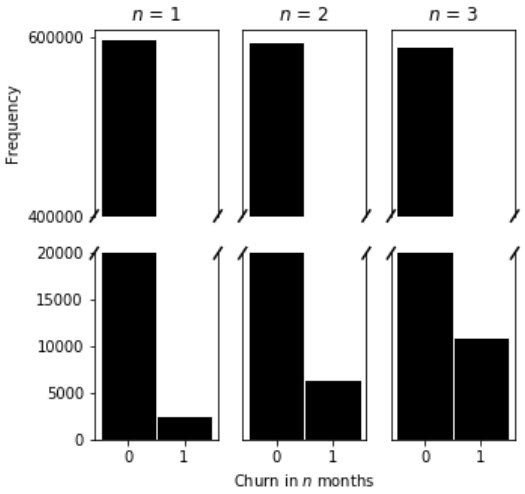


Figure 3.6: The three dependent variables: Churn in one, two and three months.

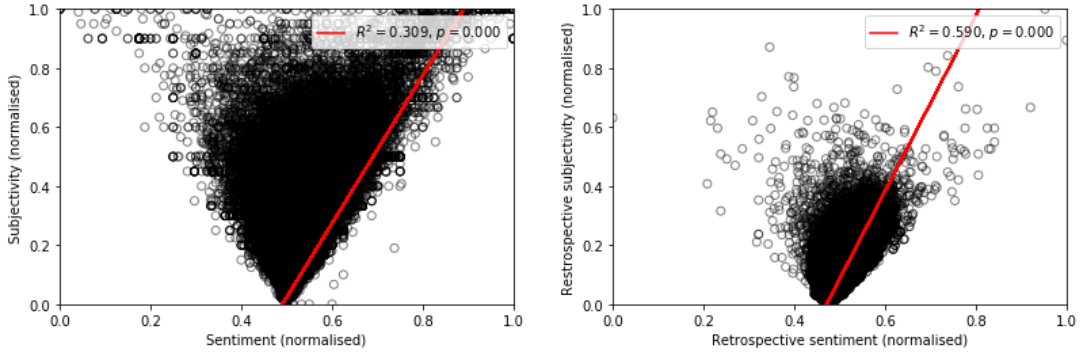
number of independent variables, d is the number of dependent variables, and r is the number of variables that is selected for a combination (a value between 1 and n). With $n = 12$, and $d = 3$, the number of different feature combinations (and hence models to train and compare) is 4755.

$$d * \sum_{r=1}^{n} \frac{n!}{r!(n-r)!)} \tag{3.1}$$

Due to limited time and computational power being available, the decision was made to group the independent variables together into five feature groups. This reduced the total number of models to train and compare to 93 ($n = 5$, $d = 3$). The feature groups were chosen manually, based on the similarity of the concepts the individual features represent.

The two features measured with Pattern, the tool for opinion mining, were grouped together, and will be called the 'opinion variables' (O). Both sentiment and subjectivity describe the internal state of the user, based on the same words in every post. Together, they describe the opinion state of the user in a two-dimensional space: recall Figure 3.2, which described the triangular relationship between sentiment and subjectivity. Figure 3.7 shows how the normalised sentiment and subjectivity variables correlate, and shows that the triangular relation between sentiment and subjectivity also holds in the data. The correlation between sentiment and subjectivity is significant ($p < 0.001$), both for the static ($R^2 = 0.31$) and retrospective ($R^2 = 0.59$) features.

The post length of a feature is expected to be a good predictor of churn: users who write longer posts are more engaged with the forum and are expected to stay longer. Both the feature sentence length and questions depend on the feature post length. Both



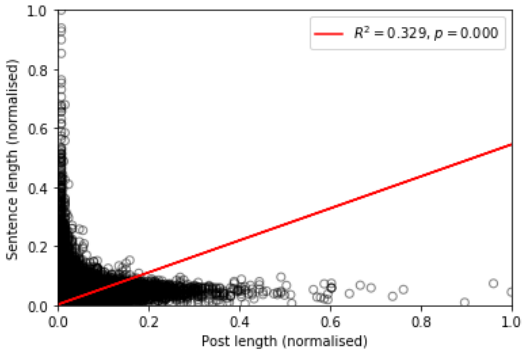
(a) The correlation between the static features (b) The correlation between the retrospective features

Figure 3.7: the two-dimensional space of opinion mining. Also the correlation between the normalised sentiment and subjectivity variables. Significance (p) and R^2) of the correlation are shown in the upper right corner.

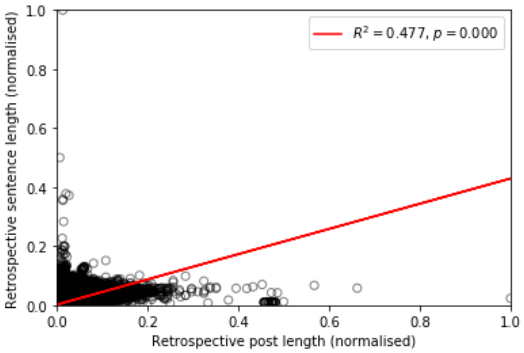
these features divide some count c by the post length. For sentence length, c = the sum of sentence lengths, and for questions, c = the sum of questions. These features were therefore grouped into one feature group, which will be called the group with 'textual features' (T). Figure 3.8 shows how the three features were correlated in the data. All correlations are significant ($p < 0.001$). The R^2-values are shown in the upper right corner of every figure.

Finally, static and retrospective features were split into different groups, such that the expected difference in performance would show clearly. An exception was made for the static and retrospective inactivity variables, as they were observed to be highly correlated. Figure 3.9 contains the correlation plot of the two normalised variables. It shows that the correlation is significant, and only has very little variation ($p < 0.001$, $R^2 = 0.97$).

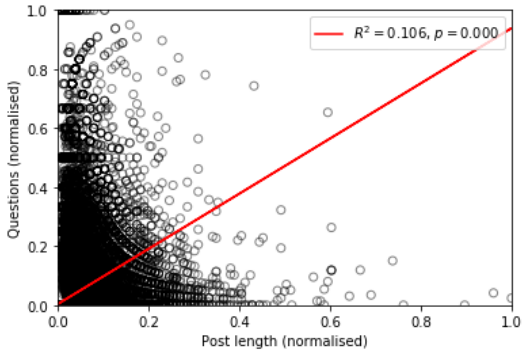
The result is five different feature groups. They are O: opinion features, T: textual features, O_r: retrospective opinion features, T_r: retrospective textual features, and I: inactivity features.
In summary: The group with 'inactivity variables' (I), contains both the static and retrospective Inactivity features. The group with 'opinion variables' (O) contains the two static variables describing an author's stance towards a subject: Sentiment and Subjectivity. Its counterpart O_r contains the retrospective counterparts of the Sentiment and Subjectivity variables. The group with 'textual variables' (T) contains the three static variables that describe the post on a textual level: Post_length, Sentence_length, and Question_marks. Its counterpart T_r contains the retrospective counterparts of the Post_length, Sentence_length, and Question_marks variables. Table 3.4 shows the five feature groups, their abbreviations, and the features the groups contain for future refer-



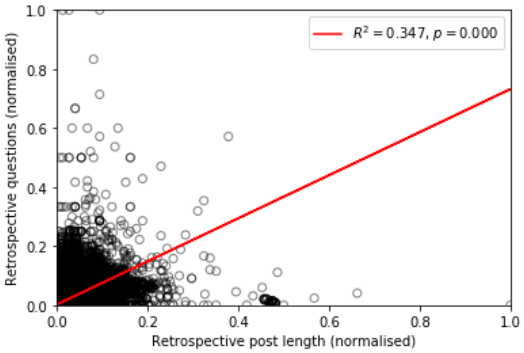
(a) Static post and sentence length



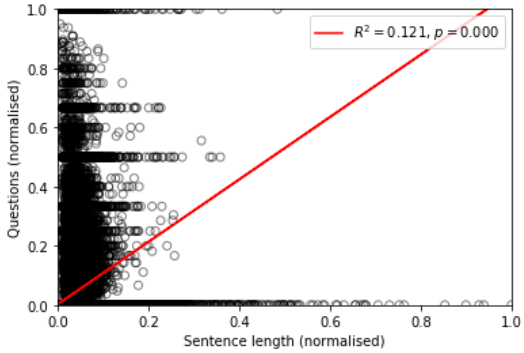
(b) Retrospective post and sentence length



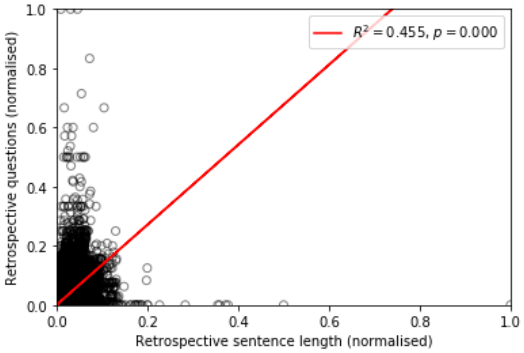
(c) Static post length and questions



(d) Retrospective post length and questions



(e) Static sentence length and questions



(f) Retrospective sent. length and questions

Figure 3.8: The correlations between textual features post length, sentence length and questions. Significance (p) and R^2 of the correlation are shown in the upper right corner.

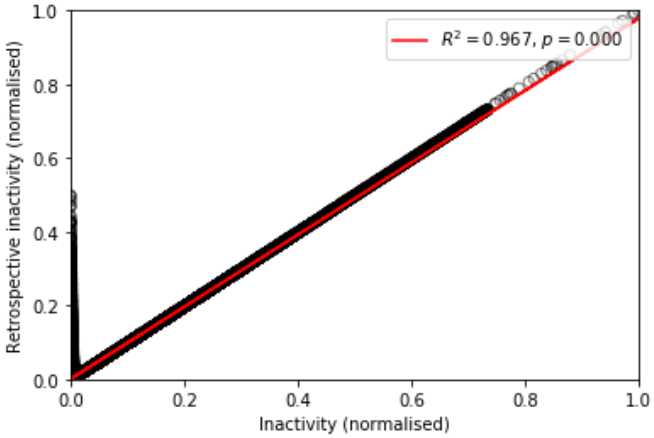


Figure 3.9: The correlation between the normalised static and retrospective inactivity variables. Significance (p) and $R^2$of the correlation are shown in the upper right corner.

ence.

The 93 different combinations of features that were made with these five groups, are used to train and test models in Section 4.1. The results of these experiments are used to answer the main questions that this thesis addresses: which dependent variable can be predicted most accurately; and which features predict patient churn most accurately, one, two, and three months into the future?

Table 3.4: The five feature groups, and the features they contain

	Feature group	**Features**
I:	Inactivity variables	Inactivity, Inactivity_mean
O:	Opinion variables	Subjectivity, Sentiment
T:	Textual variables	Sentence_length, Post_length, Question_marks
O_r:	O, retrospective	Subjectivity_mean, Sentiment_mean
T_r:	T, retrospective	Sentence_length_mean, Post_length_mean, Question_marks_mean

Chapter 4

Experiments

In order to determine how accurately churn can be predicted with the supervised machine learning algorithm XGBoost, and which features are the most important predictors, several experiments were conducted. The imbalance of the data was handled in two different ways (by weighting and balancing). This section describes how the experiments were performed.

The complete procedure is shown in Algorithm 1. Dataset $\mathcal{D}$ was built by annotating the twelve independent variables, and three dependent variables as described in Chapter 3. After this, the procedure is as follows: First, the dataset is split into k random folds, such that $D_k \subset \mathcal{D}$. The fold is shrunk to 5% of its original size when 'weighting' is chosen as the method to handle the imbalance of the samples (negative s^0 and positive s^1) in the dataset. Otherwise, all samples are taken to the next step in the algorithm.
Each fold is then split into three subsets: a training set D_k^{train}, which contains 60% of the samples in D_k, a development set D_k^{dev}, which contains 20% of the samples in D_k, and a test set D_k^{test}, which also contains 20% of the samples in D_k. As leakage of information between the sets is undesired (for it decreases the validity of the model), it is important that these sets do not overlap. All samples are in one set, and in one set only. In other words, D_k is equivalent to the disjoint union ($\sqcup$) of the three subsets.
At this point, one of the dependent variables $i \in y^{\mathcal{D}}$ is selected, as well as a set of independent variables G, which is one of all possible combinations (as calculated with Equation 3.1) $\mathcal{G}$. All independent and dependent variables in D_k^{train},D_k^{dev}, and D_k^{test} are removed, such that only those (i, g), where $g \in G$, remain to train and test the model on. These subsets are notated as $D_k^{train}(C), D_k^{dev}(C), D_k^{test}(C)$, where $C == (i, g)$. If the imbalance of the data is handled by 'balancing', then the number of positive and negative samples in the dependent variables is made equal at this point: $|s^0| == |s^1|$, for all three subsets $D_k^{train}(C), D_k^{dev}(C)$, and $D_k^{test}(C)$.
Finally, grid search (Algorithm 4) is performed on the train and development set to determine the best set of parameters, and the model is trained and tested on the train and test set using the best parameter settings (Algorithm 2).

Algorithm 1 Main code: calculating AUC scores for all possible combinations of dependent variables and feature groups, while handling class imbalance.

Input: A normalised dataset $\mathcal{D}$, split into dependent variables $y^{\mathcal{D}}$ and independent variables $X^{\mathcal{D}}$, sorted into feature groups $\mathcal{G}$; The number of folds $k \geq 2$; Parameters for unbalanced data *balance* $\in \{\top, \bot\}$ and *weight* $\in \{\top, \bot\}$; All possible hyper-parameter settings $\mathcal{P}$;
Output: A set of *AUC* ROC-AUC scores: k for every possible combination of dependent variables and feature groups

Split $\mathcal{D}$ into k random folds: $D_k \subset \mathcal{D}$
if *weight* $== \top$ **then**
    $d_k \leftarrow$ random samples from D_k, such that $|d_k| == |\binom{D_k}{0.05|D_k|}|$
    $D_k \leftarrow d_k$
end if
for $D_k \subset \mathcal{D}$ **do**
    Split D_k into a train set (60%), development set (20%) and test set (20%),
    such that all samples are uniquely in one set: $D_k == D_k^{train} \sqcup D_k^{dev} \sqcup D_k^{test}$
    for dependent variable $i \in y^{\mathcal{D}}$ **do**
        for independent variable group $G \in \mathcal{G}$ **do**
            Remove those variables from the train, development and test sets,
            such that the model is trained, developed, and tested on features $(i, g \in G)$.
            With C as combination (i, G): $D_k^{train}(C), D_k^{dev}(C), D_k^{test}(C)$
            if *balance* $== \top$ **then**
                Balance the positive (1) and negative (0) samples $s \in y^{D_k^{train}}(C)$,
                such that $|\{s^1 \in y^{D_k^{train}(C)}\}| == |\{s^0 \in y^{D_k^{train}(C)}\}|$
                Balance s^1 and s^0 in $y^{D_k^{dev}(C)}$, and $y^{D_k^{test}(C)}$ in the same way
            end if
            $bestP \leftarrow \max(\texttt{GridSearch}(D_k^{train}(C), D_k^{dev}(C), P, weight))$ ▷ Alg. 4
            $AUC^{(C)} \leftarrow \texttt{XGBoost}(D_k^{train}(C), D_k^{test}(C), bestP, weight)$ ▷ Alg. 2
        end for
    end for
end for

Section 4.1 will provide an in-depth overview of the XGBoost algorithm. Section 4.3 will motivate the choices that were made in the selection of hyper-parameters, and Section 4.2 will go deeper into the two methods that were evaluated to handle the class imbalance with. In addition, Figure 4.1 gives a graphical representation of the basic procedure. Measures to handle class imbalance (weighting, balancing) are not shown in the figure.

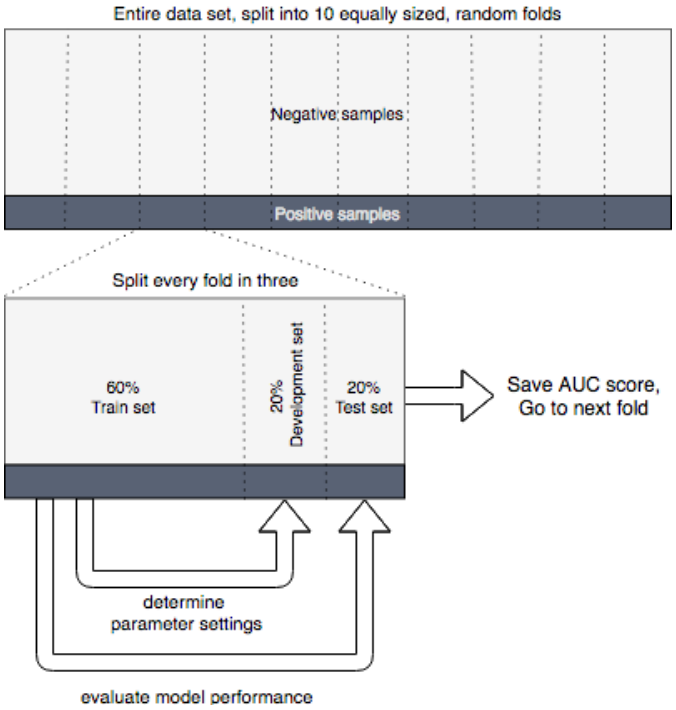


Figure 4.1: A simplified graphical representation of the experiments, that shows how the ten AUC scores for every combination of dependent and independent variables were obtained.

4.1 The XGBoost algorithm

Gradient boosting was first proposed by Friedman (2001). This principle was modified for decision trees, which Chen and Guestrin (2016) used to create gradient boosted decision trees. They called their algorithm 'XGBoost'. This supervised Machine learning algorithm has proved itself in several Kaggle competitions. Moreover, the algorithm "runs more than ten times faster than existing popular solutions on a single machine" (Chen and Guestrin, 2016, p. 785). This section will dive into the workings of XGBoost.

Imagine that you have a question. You ask your neighbour, who gives you a certain answer. Their answer by itself is not a strong indicator of the real answer. Therefore, you decide to ask 100 others: you ask the other neighbours, your parents, some tourists ... By themselves, their answers are only weak indicators of the actual answer to your question. However, when you combine their answers, you can be more and more sure of what the actual answer to your question is.
This is an example of an *ensemble* method: many weak predictors are combined, which result in one strong predictor. XGBoost is also an ensemble method. Many different decision trees are built on the data, which by themselves do not have a large predictive value, but combined do. Moreover, XGBoost uses gradient boosting: with this method,

Algorithm 2 "XGBoost": How to train and test a model with a given parameter setting.

Input: A train set $D^{train} == D_k^{train}(C)$; A development set $D^{dev} == D_k^{dev}(C)$ or $D_k^{test}(C)$; A parameter setting $p \in \mathcal{P}$; Parameter for unbalanced data $weight \in \{\top, \bot\}$
Output: An AUC score

if $weight == \top$ **then**
    $ratio \leftarrow \frac{|s^0|}{|s^1|}$,
    where s^0 and s^1 are the positive and negative samples $\in y^{D^{train}}$, respectively
end if
$paramSettings \leftarrow p \cup ratio$
$classifier \leftarrow$ train XGBoostClassifier() with $paramSettings$ on D^{train}.
$\hat{y} \leftarrow$ predict dependent variable y of D^{dev}
$AUC \leftarrow$ calculate the AUC for $(y, \hat{y})$
return AUC

decision trees are built sequentially ('on top of each other'), minimising a given loss function.

Every single dicision tree divides the dataset into smaller groups. These groups are the result of splits that were made in some of the features. Every sample (data point) that occurs in the data travels through the tree from top to bottom, and at every node the sample's feature value is compared to a threshold. Depending on whether the sample's value is over or under the threshold, it is sent to either the left or right child node. These steps are repeated until the leaf node is reached at the maximum depth of the tree. All samples that end up in the same leaf node are called the instance set of the leaf node. In addition, the leaf node contains a weight value, which determines which prediction should be made for the samples in the instance set. A single decision tree is defined as:

$$f(x) = w_{q(x)} \tag{4.1}$$

In this equation, $q(x)$ denotes the instance set which sample x belongs to, and $w_{q(x)}$ denotes the weight of the child node. A decision tree is thus defined as a function of the weights of the nodes containing the instance sets.

With respect to predictions, some trees divide the data in more useful groups than others. This is directly reflected in the predictive accuracy of a single decision tree. However, XGBoost trains a gradient *boosted* tree ensemble, which means that decision trees are added to an ensemble, sequentially: the 'usefulness' of the next model that is added to the ensemble, is therefore dependent on the trees that the ensemble already contains. Specifically, the next tree that is added must contribute as much as possible to the ensemble. The *objective function* is used to determine which tree satisfies this condition.

The objective function is used in supervised learning problems to optimise certain model behaviour. In the case of XGBoost, the objective is to make a model with the highest predictive accuracy possible, while at the same time keeping the model simple.

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta) \tag{4.2}$$

In Equation 4.2, $L(\Theta)$ is a loss function. It describes how well the model fits the training data. $\Omega(\Theta)$ is a regularisation function. It describes the complexity of the model.

In XGBoost, the loss function $L(\Theta)$ is decomposed over the individual samples. The sum of all these *local objectives* is then combined with the regularisation function $\Omega(\Theta)$. The regularisation function is the sum of the complexity of all trees. Together, these sums make up the *global objective* of the entire dataset.

$$\mathcal{L} = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{4.3}$$

In Equation 4.3, $\ell(y_i, \hat{y}_i)$ denotes the local objective. It is calculated for every i^{th} sample in the dataset. $\Omega(f_k)$ denotes the complexity of every k^{th} tree in the ensemble. Together, these two sums define the global objective $\mathcal{L}$.
In the local objective, y denotes the true labels (dependent variable values) of the samples in the train set, and $\hat{y}$ denotes the predicted labels. For binary classification, the local objective is equal to $\ell(y, \hat{y}) = \log(1 + \exp((1 - y)\hat{y})$ (Cho, 2017). In the regularisation function, f_k denotes a tree k, and Ω is a function that penalises its complexity.

XGBoost selects the next tree to be added to the ensemble, such that by adding this tree f_t to the existing ensemble of $(t - 1)$ trees, the global objective is minimised:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} \ell(y_i + \hat{y}_i^{(t-1)} + f_t(x_i)) + \sum_{k=1}^{K} \Omega(f_k) \tag{4.4}$$

All constant terms are then dropped, second-order-approximation is used to find the minimal value for the objective, and the result is expressed in terms of items in the instance set, instead of samples in the data set. This 'simplified objective' Cho (2017) is denoted with $\tilde{\mathcal{L}}^{(t)}$ and expressed as follows:

$$\tilde{\mathcal{L}}^{(t)} = -\frac{1}{2} \sum_{j=1}^{T} \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \tag{4.5}$$

Where T is the number of instance sets (leaf nodes), I_j is an instance set of node j, and g_i and h_i are partial derivatives of the loss function corresponding to every i^{th} element in the instance set. λ and γ control how much influence the local objective has, and how heavily model complexity is penalised.

Algorithm 3 Selecting the tree that contributes to the ensemble as much as possible. Adapted from (Cho, 2017, p. 4)

Input: The train data $\mathcal{D} = \{s_1, \dots s_n\}$; Regularisation parameters λ and γ; the set of all tree structures $\mathcal{Q}$, defined over n samples.
Output: the tree structure q_t that contributes to the ensemble as much as possible.

for $q \in \mathcal{Q}$ **do**
  $T \leftarrow$ [leaves defined by q]
  for leaf $j \in T$ **do**
    Instance set $I_j \leftarrow \{i | q(s_i) = j\}$
    Sum of gradients $G_j \leftarrow \sum_{i \in I_j} g_i$, where g_i is the gradient of i
    Sum of hessians $H_j \leftarrow \sum_{i \in I_j} h_i$, where h_i is the hessian of i
  end for
  $\tilde{\mathcal{L}}^{(t)}(q) \leftarrow -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$ ▷ Equation 4.4
end for
$bestQ \leftarrow q \in Q$ that minimises $\tilde{\mathcal{L}}^{(t)}(q)$
return $bestQ$

Algorithm 3 summarises this procedure.

Most important to notice about the simplified objective, is that the global loss function is now completely dependent on the sums of partial derivatives of the loss function. This is why the algorithm is a *gradient* boosting algorithm: it uses gradient descent to find the lowest value of the loss function. One benefit of this method, is that the algorithm focusses more on incorrectly classified samples in the next iteration round. This is especially beneficial in the case of imbalanced data, because the minority class is typically harder to predict than the majority class, causing the algorithm to focus extra on samples from the minority class.
In order not to 'overshoot' the minimum value of the loss function, every tree that is added to the ensemble is multiplied by the *learning rate*. The smaller the learning rate, the more accurately the minimum can be determined. However, reaching the minimum will also take longer. The ensemble is complete, when the maximum number of rounds is reached, or when it starts to overfit the training sample. When the predictive accuracy on the train set keeps increasing, but the predictive accuracy on the test set starts decreasing, this is a sign that the model is overfitting. Stopping the algorithm before this point is an important measure against overfit.

Fortunately, the implementation of Chen and Guestrin (2016)'s algorithm for XGBoost in Python includes multiple measures to improve model performance, and reduce overfitting. In order to determine the right combination of these measures, a grid search is performed. In Section 4.3, the selected hyper-parameters to be used by grid search are motivated. Section 4.2 describes the two methods that were compared to handle the

problem of class imbalance.

4.2 Training and testing the models

The complete data set, which has a total of 601892 data points, is severely imbalanced for all three dependent variables. The ratio of positive:negative samples for churn in one month is (1:239), for churn in two months (1:94), and for churn in three months (1:55). The positive class of all three dependent variables is *relatively rare*, as Weiss (2015) calls it, compared to the negative class. If this kind of data is handled inappropriately, data mining techniques are likely to perform badly. Machine learning models trained on this data will often be biased towards the majority set. Weiss points out that in extremely imbalanced data sets, it may even occur that the algorithm is unable to formulate classification rules for the minority class, resulting in a recall of 0. This means that none of the samples from the minority class is classified correctly.

Two approaches were tried and compared to deal with this problem of relative rarity. Both approaches are first summarised shortly, then motivated and explained more elaborately.

1. *Weighting the data*: All 601892 samples were preserved, but the positive samples (samples from the minority class) were given a higher penalty when they were misclassified during training. *Because this method, in contrast to balancing, preserved all samples in the data set, a random subset of the data was selected to decrease runtime.*

2. *Balancing the data*: All positive samples were preserved (2517 for churn in one, 6405 for churn in two, and 10976 for churn in three months), and exactly as many negative samples were randomly selected from all negative samples. *Because testing the model on a balanced test set is not representative of the model's behaviour in a real-world situation, a variation with an unbalanced test set was evaluated as well.*

4.2.1 Weighting the data

As explained in section 4.1, XGBoost is an iterative algorithm. It trains an ensemble of trees, and during every iteration tries to improve the predictive accuracy of the ensemble. Wrongly classified samples s decrease the predictive accuracy of the model, and are therefore, thanks to gradient boosting, focussed on extra in the next iteration.
This by itself is beneficial in the case of imbalanced data, as the minority class is often harder to predict than the majority class and thus gets more attention from the algorithm. In addition, XGBoost contains the parameter `scale_pos_weight`. With this parameter, the penalty for wrongly predicting samples from the minority class can be increased, forcing XGBoost to focus even more on these samples. This method allows the algorithm to learn from all samples in the data set, while making sure that the minority

class is not easily overlooked. The chosen value for this parameter of XGBoost was the ratio negative:positive samples ($\frac{|s^0|}{|s^1|}$) in the train set (see Algorithm 1, lines 1-3).
A random sample with the original ratio positive:negative samples was made from the dataset, in order to decrease the runtime of the algorithm. Of all 601892 samples, about 30000 (5%) samples were used (see Algorithm 1, lines 2-4). The number of folds was initially set to $k = 10$, but the results, which will be presented in Chapter 5, showed that the predictions were inconsistent. In an effort to improve the model's predictions, the experiments were re-run, but with $k = 5$.

4.2.2 Balancing the data

Balancing of unbalanced data is a popular way to handle the relative rarity of a class. There are two approaches that can be taken: One can either *increase* the number of samples in the minority class (copying existing samples, or using more advanced techniques, see Weiss), or *decrease* the number of samples in the majority class. Because over-sampling increases the total number of samples, and thereby increases the total runtime of the algorithm, under-sampling was used to balance the data. In its simplest form, under-sampling is the random removal of samples in the majority class, until the majority class is the same size as the minority class. Burez and Van den Poel (2009) for example used this technique on customer churn data very similar to the data used in this thesis. Moreover, they show that under-sampling can lead to an improved prediction accuracy, especially when the results are evaluated with ROC-AUC. Using more advanced under-sampling techniques did not significantly improve the prediction accuracy.
The simplest form of under-sampling was used to balance the classes for this thesis as well. This method allows the algorithm to learn as much about positive samples in the data set as about negative samples in the data set, because both are equivalently present. On the downside, many potentially useful data points were discarded, which may decrease the performance of the model.

The three dependent variables all have a different ratio of positive:negative samples ($\frac{|s^1|}{|s^0|}$). Therefore, the total number of samples s in the data, depends on the dependent variable i that is to be predicted. With i = churn in one month, the data set contained 2517 s^1. With i = churn in two months, the data set contained 6405 s^1. With i = churn in three month, the data set contained 10976 s^1. In Algorithm 1, balancing occurs in line 14-17. It shows that the train set D_k^{train}, development set D_k^{dev} and test set D_k^{test} were balanced in every k^{th} fold. Training, developing and testing was thus done on balanced sets.

To get a better indication of how useful the models would be in a realistic setting, the models were also developed and tested on unbalanced data. For these 'unbalanced' experiments, the exact same k folds were used as for the balanced experiments. Any observed difference in performance between the two models can thus only be explained

by the (im)balance of test set D_k^{test}. The performance of the models that predicted each of the dependent variables most accurately is finally further analysed by looking at the false negative rate (FNR). The false negative rate can be understood as the number of times that a user was about to churn within one, two or three months, but was not recognised as such. It is calculated by dividing the number of falsely predicted negative samples (FN; the unrecognised churners) by the total number of positive samples (FN + truly predicted positive samples (TP); all churners). All experiments were run with $k = 10$.

4.3 Hyper-parameter selection

XGBoost allows adjustment of several parameters. Choosing the wrong set of parameters can result in an underfitted or overfitted model, so choosing the parameters correctly is important. The optimal setting of the parameters depends on the data that is available. Therefore, the parameter settings were optimised for every unique variable combination (i, G), where i is the dependent variable, and G is a selection of independent variables. The parameter settings were optimised with grid search. Grid search is an exhaustive search through a large set of possible parameter combinations. As Algorithm 4 shows, for every parameter combination, a model is built on the train set $\mathcal{D}^{train}$, and tested on the development set $\mathcal{D}^{test}$. When all parameter settings are evaluated, the parameter settings that boosted the performance measure (in this thesis: the Area Under the Receiver Operating Curve, or AUC for short) highest are the optimal parameter settings (Algorithm 1, line 19). These are used to train the model one last time on the train set. The performance of this model is then evaluated on the test set (Algorithm 1, line 20). This was done for all combinations of dependent and independent variables.
This process was repeated k times: once for every fold. This resulted in k AUC scores for every combination (i, G). For every combination the average AUC over the folds was calculated, as well as the standard deviation. This reflected the model's predictive accuracy.

This section presents an exploration of the influence of different hyper-parameters on the predictive accuracy. A selection of parameters to be used in grid search was made, based on this exploration. An overview of the grid search procedure is given in Algorithm 4, and Algorithm 1 shows how grid search is embedded in the main procedure.

Table 4.1 shows the parameters for the XGBoost algorithm for classification problems. To determine which parameter settings produce the best predictions, their influence on the workings of XGBoost was examined, and on a selection of them, grid search was performed. This exploration revealed which hyper-parameters most effectively tuned the model performance. A selection of values for these parameters was made, which was used as input P in the main procedure.

The results of the explorations that are presented in this section, were obtained with

Algorithm 4 "GridSearch": How to determine the optimal parameter settings

Input: A train set $D^{train} == D_k^{train}(C)$; A development set $D^{dev} == D_k^{dev}(C)$; All possible parameter settings $\mathcal{P}$; Parameter for unbalanced data $weight \in \{\top, \bot\}$
Output: $AUC^{\mathcal{P}}$, A set of ROC-AUC scores, one for every setting $p \in \mathcal{P}$

1: $AUC\mathcal{P} = \{\}$
2: **for** $p \in \mathcal{P}$ **do**
3: $\cup AUC^p \leftarrow$ `XGBoost`$(D^{train}, D^{dev}, p, weight)$
4: $AUC^{\mathcal{P}} \leftarrow AUC^{\mathcal{P}} \cup AUC^p$
5: **end for**
6: **return** $AUC^{\mathcal{P}}$

Table 4.1: The default settings of the XGBoost classification algorithm (`XGBClassifier()`)

base_score	booster	colsample_bylevel	colsample_bytree	gamma
0.5	'gbtree'	1	1	0
learning_rate	max_delta_step	max_depth	min_child_weight	missing
0.1	0	3	1	None
n_estimators	n_jobs	nthread	objective	random_state
100	1	None	'binary:logistic'	0
reg_alpha	reg_lambda	scale_pos_weight	seed	silent
0	1	1	None	True
subsample				
1				

a 5-fold cross validated grid search on all samples of the data set. As dependent variable, 'churn in three months' was chosen, and the complete set of independent variables (I, O, T, O_r, T_r) was used to predict it with. The data was balanced. To make sure that differences observed between different explorations are explained by the different parameter settings, the same random seed was used for every exploration.

Table 4.1 shows the default settings of the XGBoost classification algorithm. The (potential) use of all of these parameters is described in the XGBoost documentation (Distributed (Deep) Machine Learning Community, 2015), and also presented here with the necessary adaptations. Whenever possible, the parameters are linked to concepts described earlier in the thesis. For all parameters in Table 4.1, from left to right, top to bottom (alphabetical order):

XGBoost builds trees sequentially, choosing the next tree by optimising the objective. However, what is there to optimise, when there are no trees in the ensemble yet? The solution is simple: predict all dependent variables the same, and improve that score in the next round. This initial prediction is called the `base_score`, or global bias. Usually, varying this parameter does not influence the predictive accuracy of the model

significantly.

XGBoost can be used with different `booster`s. In this thesis, gradient boosted trees were used, but it is also possible to use a linear model[1].

The parameter `colsample_bylevel` can be set anywhere in between 0 an 1 to reduce overfitting. XGBoost then becomes similar to random forests: At every node in the tree, only a part of the independent features is considered for the next split.

`colsample_bytree` is similar, but is used to determine the ratio of dependent features that may be used in every tree.

Another parameter to reduce overfitting is `gamma`. Gamma is the minimum required loss to make a leaf node into a parent node. A high value can reduce overfitting of the model, as the dataset is split into less subgroups and thus stays more generalisable.

In order not to overshoot the minimum value of the loss function, the influence of every new tree is smaller than the last. To obtain this, the error residuals of every sequential tree are multiplied by the `learning_rate`, which can be anywhere between 0 and 1.

The `max_delta_step` controls the maximum weight update that the algorithm is allowed to make in every round. The step difference (delta step) is an important element in the simplified objective $\tilde{\mathcal{L}}^{(t)}$, and is the sum of gradients G_j divided by the sum of hessians H_j plus the regularisation parameter λ (see Algorithm 3): $\frac{G_j}{H_j+\lambda}$. Recall that the weight associated with an instance set determines the prediction that is made for all samples in that instance set.

Another maximum is the `max_depth` of a tree. This value refers to the maximum number of allowed splits. A high value can increase the predictive accuracy of the model, but also makes it more likely to overfit.

Just like gamma, the parameter `min_child_weight` can reduce overfitting of a model by preventing a new split. However, where gamma required a minimum loss obtained by the split, this parameter requires a minimum weight for the resulting child node.

The parameter `missing` is used to specify which values in the data are missing values (for example: 0, ‘’, or -1). During training, XGBoost learns which direction is the best direction to send all missing values.

The number of estimators, `n_estimators`, refers to the number of models that may be built sequentially. Without the proper counter-measures, setting this number too high can lead to overfitting. However, when fitting the model, XGBoost also uses the additional parameter early_stopping_rounds. This parameter stops the algorithm if the predictive accuracy on the test set has not increased for a number of rounds. It is therefore often the case that the algorithm is stopped before n_estimators is reached.

XGBoost builds decision trees sequentially. Because the decision of the next ‘best model’ depends on the models that are already contained in the ensemble, decision trees can not be built in parallel. However, the different branches of a tree can be built in parallel. This is used to speed up the algorithm. With `n_jobs`, the number of cores that the algorithm may use for the computation can be set.
It used to be called `nthread`, which is now deprecated.

[1] https://github.com/dmlc/xgboost/blob/master/R-package/demo/generalized_linear_model.R

The `objective` defines the loss function $\ell y, \hat{y}$ that needs to be minimised. As churn prediction in this thesis was handled as a binary classification problem, the built-in objective function called 'binary:logistic' was used. it uses the function $\ell y, \hat{y} = \log(1 + \exp((1-y)\hat{y}))$ to return the predicted probability of a sample, *not* the class Cho (2017).

XGboost can perform both LASSO regression and Ridge regression to reduce overfitting of the model. LASSO is called `reg_alpha` in the parameter table, and Ridge is called `reg_lambda`. These two correspond to the two regularisation terms λ and α, respectively, and together control the model complexity. Raising reg_lambda will decrease the influence of some of the features, while increasing reg_alpha will even completely remove some features.

The parameter `scale_pos_weight` can be used to weight the data in the case of class imbalance, as described in Section 4.2. It punishes the algorithm more severely for wrongly predicting samples from the minority class, than for wrongly predicting samples from the majority class.

The random `seed` seed can be set to control randomness. Controlling randomness makes it easier to compare results between runs, as differences can then not be explained by random variation any longer. Still, it is possible that different runs cause slightly different results, "due to non-terminism in floating point summation order and multi-threading" (Distributed (Deep) Machine Learning Community, 2015, FAQ section).
It used to be called `random_state`, which is now deprecated.

The next parameter is more of an aesthetic one: setting `silent` to 0 suppresses written output from the algorithm, while setting it to 1 (or higher) enables the algorithm to print out intermediate results.

Finally, XGBoost can not just sample columns (features) with the colsample parameters, but also rows (samples, data points) with the `subsample` parameter. A value between 0 and 1 is specified to determine the ratio of all samples that is classified in the next iteration round. Lower values reduce overfitting, but too low values may cause underfitting.

Parameters that were unnecessary to explore in grid search, based on their documentation, were base_score (0.5), booster ('gbtree'), missing (None), n_jobs (although it was set to -1 to enable XGBoost the use of all available cores), nthread (deprecated), objective ('binary:logistic'), scale_pos_weight (1), seed (which was set to 42), random_state (deprecated) and silent (True). These settings remain constant throughout all explorations presented in this section. The other values are explored below.

First, the model was trained with the default settings summarised above. The results showed that the predictive accuracy of the model trained with default values, expressed in (ROC-)AUC, was 0.691 ± 0.007.

μ AUC	σ
0.694	0.007

The learning rate of the model was then varied. It was increased, as well as decreased,

compared to the default settings. The results showed that the model had the highest predictive accuracy when the highest learning rate was used(with learning_rate=0.2, 0.736 ± 0.009). This indicates that the number of rounds (n_estimators) was set too low for the algorithm to reach the maximum ROC-AUC score.

learning_rate	μ AUC	σ
0.2	0.736	0.009
0.1	0.695	0.006
0.05	0.666	0.004

Therefore, the next logical step was to increase n_estimators. It was increased to 800. The results showed a large increase in predictive accuracy. Although the best performance was still with the highest learning rate (with learning_rate = 0.2, 0.822 ± 0.005), the decision was made to keep the learning rate at 800. The mean AUC score is already very promising, and other variables will be tuned as well.

n_estimators	learning_rate	μ AUC	σ
800	0.2	0.822	0.005
800	0.1	0.799	0.008
800	0.05	0.766	0.009

Next, the maximum depth of the trees was varied. A range of depths was explored: 4, 6, 8, 10, and 12. The results showed that tree depths of 8 and up performed best. Using deeper trees (than the default 3) also decreased the optimal learning rate, indicating that the model was able to get closer to the optimum with these settings. The highest AUC score was reached with max_depth = 8, learning_rate = 0.1 (0.851 ± 0.006). The top three results are shown below.

max_depth	n_estimators	learning_rate	μ AUC	σ
8	800	0.1	0.851	0.006
10	800	0.1	0.850	0.006
10	800	0.05	0.850	0.006

For the next exploration, the parameter min_child_weight was added to the grid search, with values 1, 2, 4, 6, and 8. The results showed that with learning_rate = 0.1, max_depth = 8, min_child_weight = 1 the highest AUC score was reached (0.849 ± 0.004).

min_child_weight	max_depth	n_estimators	learning_rate	μ AUC	σ
1	8	800	0.1	0.849	0.004
2	8	800	0.1	0.847	0.005
1	10	800	0.05	0.847	0.005

As this was the default setting, and other results showed that higher values only resulted in a lower AUC score, this parameter was not included in further explorations. For the last exploration, a wide spread of values for gamma was added to the grid search: 0, 0.1, 1, 10 and 100. The results showed that with learning_rate = 0.1, max_depth = 10, gamma = 0.1, the highest AUC score was reached (0.851 ± 0.004).

gamma	max_depth	n_estimators	learning_rate	μ AUC	σ
0.1	10	800	0.1	0.851	0.004
0.1	8	800	0.1	0.851	0.004
0.1	8	800	0.2	0.851	0.005

However, comparing these results to the results of the grid search which used the same settings for max_depth and learning_rate but only the default setting for gamma, reveals that the maximum AUC did not increase (with max_depth = 10, learning_rate = 0.1, gamma = 0.1, 0.851 ± 0.004; with max_depth = 8, learning_rate = 0.1, gamma = 0.0, 0.851 ± 0.006).

Parameters that were not explored, were colsample_bylevel, colsample_bytree, max_delta_step, reg_alpha, reg_lambda and subsample. The reason that colsample_bylevel, colsample_bytree, reg_alpha, reg_lambda and subsample were not included in the grid search, was that these were expected to decrease the experimenter's control over sample and variable influence. The set-up of the experiments was such that the influence of specific variable groups, as well as positive and negative samples was controlled. It was expected that these parameters would distort the results, making it harder to conclude which variable groups were most influential in the prediction of churn on fora.
Finally, max_delta_step was documented very poorly, only seldom used in other research, and never motivated. It does definitely need to be explored, but the decision was made to not do that within the scope of this thesis.

To conclude: the hyper-parameters that were observed to influence the predictive accuracy of the model the most, were n_estimators, max_depth and learning_rate. For n_estimators, 800 proved to be a reasonable number of rounds. For max_depth, depths 8 and higher (10, 12) improved the predictive accuracy of the model. For learning_rate, 0.05 and 0.1 usually performed optimally, and 0.2 compensated in some cases when the optimum could not be reached within 800 rounds. These were therefore the values that were chosen to perform grid search on during the experiments described in Chapter 4, and presented in Chapter 5. Table 4.2 summarises the settings.

Table 4.2: Parameter values that were selected for grid search

Name	Description	Values
max_depth	maximum tree depth	[8,10,12]
learning_rate	learning rate	[0.05, 0.1, 0.2]
n_estimators	number of trees to fit	800

Chapter 5

Results

This section contains the results of the conducted experiments. The experiments aim to answer the question whether churn can be more accurately predicted in one, two or three months, and which features are most influential in these predictions. Additionally, the experiments were conducted with weighted, as well as balanced data, to determine how the problem of class imbalance should be handled.

This section will first contain the results of weighting the data (in 10-fold and 5-fold), then the results of balancing of the data. Balancing was initially done on the training, as well as the test set. Then the models were trained on balanced, but tested on unbalanced data to determine how good the model would perform in a 'real-world' setting.

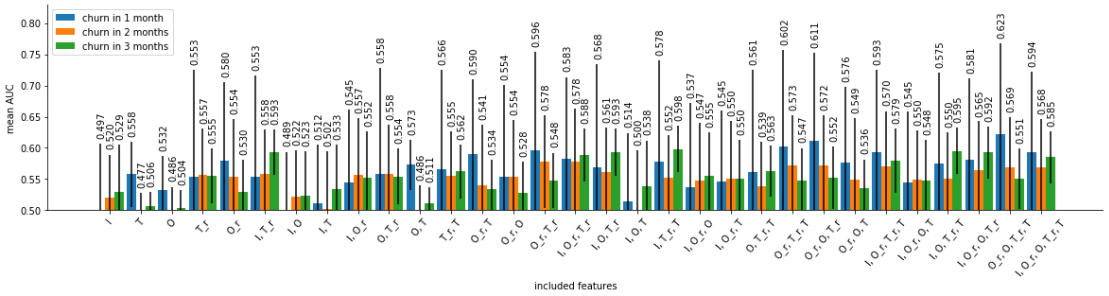
Table 3.4 is repeated as Table 5.1 for practical purposes.

5.1 Weighting the data

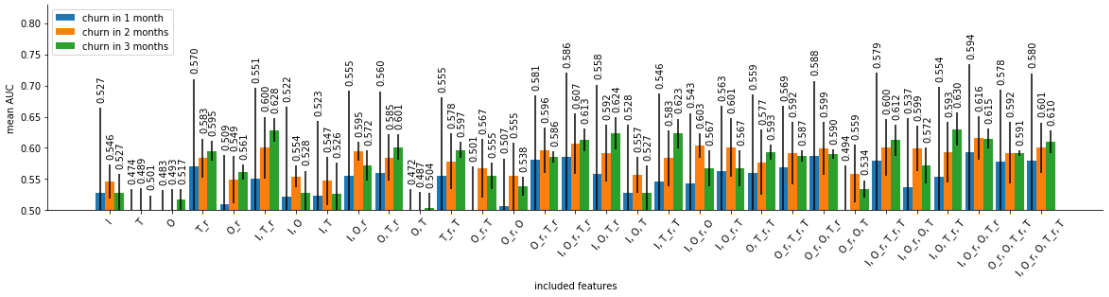
As a first effort to treat the imbalance of the data, the samples were weighted: the algorithm was punished more severely for badly predicting samples from the minority class than for badly predicting samples from the majority class. Of all data, 5% was used

Table 5.1: The five feature groups, and the features they contain (repeated)

	Feature group	**Features**
I:	Inactivity variables	Inactivity, Inactivity_mean
O:	Opinion variables	Subjectivity, Sentiment
T:	Textual variables	Sentence_length, Post_length, Question_marks
O_r:	O, retrospective	Subjectivity_mean, Sentiment_mean
T_r:	T, retrospective	Sentence_length_mean, Post_length_mean, Question_marks_mean



(a) Using 10-fold cross-validation



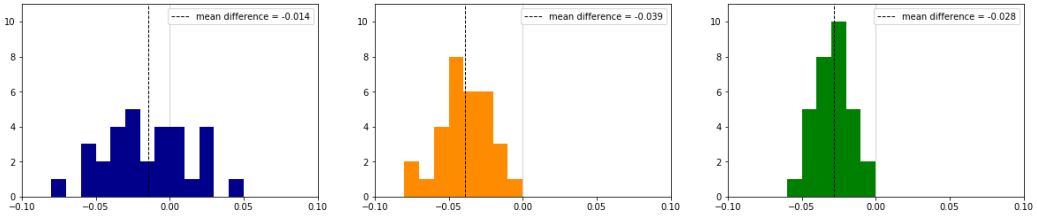
(b) Using 5-fold cross-validation

Figure 5.1: Average AUC scores of 93 models. The samples were weighted, and only 5% of the data was used in an effort to make runtime manageable

for this experiment in order to make runtime manageable. The ratio positive:negative samples was maintained, which meant that for each model, 30000 samples were used, of which 125 (for churn in one month), 320 (for churn in two months), or 545 (for churn in three months) positive samples were available. $^1/_k{}^{th}$ was used in each of k folds, of which 60% was used to test, 20% to develop, and 20% to train the model on.

The results are shown in Figure 5.1a. Without looking at the exact AUC scores, it is immediately clear that there is a lot of variation. Between different models, this variation can be very informative. However, the standard deviations (SD) of the AUC scores are high, indicating large variations *within* the models as well. Such large deviations make it hard to interpret the results of the experiment. As an effort to decrease the standard deviations, the experiment was repeated with $k = 5$ folds.

The results of the experiment done in five folds are shown in Figure 5.1b. Again, a lot of variation between and within the models is present. Figure 5.2 gives some extra insight in the differences between standard deviations in the ten- and five-fold experiments. The figure shows three histograms, one for every dependent variable. On the x-axis is the difference in SD-size, calculated as $\text{SD}^{\text{5-fold}} - \text{SD}^{\text{10-fold}}$. Negative numbers thus indicate that $\text{SD}^{\text{5-fold}}$ is smaller than $\text{SD}^{\text{10-fold}}$, for a particular model (i.e., combination of dependent and independent variables). Every histogram also shows the average



(a) With $n = 1$ (b) With $n = 2$ (c) With $n = 3$

Figure 5.2: Differences in standard deviation when predicting churn in n months: the model trained with 5-fold cross validation made less variant predictions.

difference, which is < 0 in all three: this indicates that there is indeed less variation within the experiments when $k = 5$ are used, compared to when $k = 10$ folds are used. However, there is also a lot of variation in the differences between SDs, especially when churn in one month is predicted.

Due to the large standard deviations that are present for all models, which are the result of experiments in both 5- and 10-fold, no reliable interpretation of the results can be done. Therefore, the class imbalance problem was approached in another way, namely through balancing.

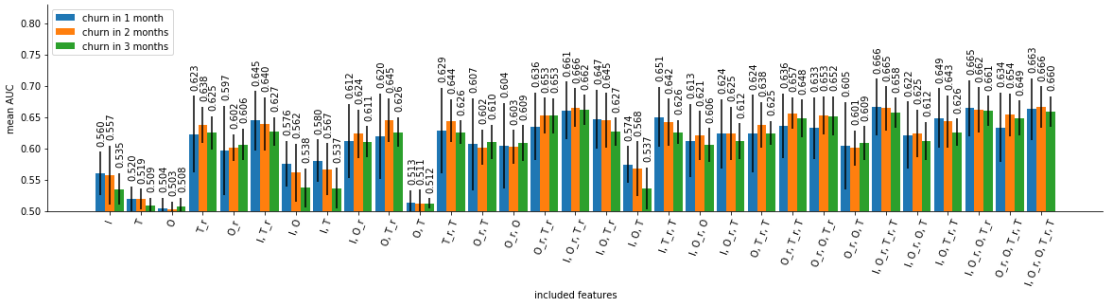
5.2 Balancing the data

The data was balanced: all n positive samples were selected, and combined with n randomly selected negative samples.

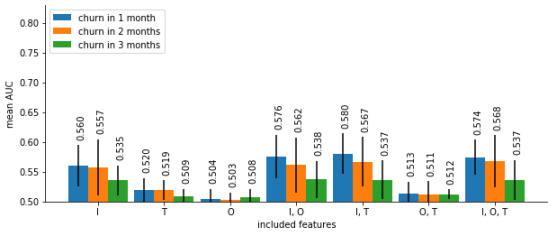
Testing on a balanced test set

Figure 5.3a shows the results of the 93 models that were trained and tested on balanced data. It immediately shows that there is less variation within the predictive accuracy of the 31 different models: the standard deviations are lower. Therefore, these results look reliable enough to examine them in some more detail to find out the influence of the different feature groups on the model's predictive accuracy. All mean AUC scores and standard deviations can be found in Appendix A. Whether two models differ significantly in performance can be looked up in Appendix B.

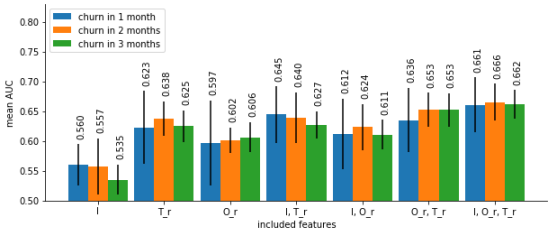
Although the variation *within* the experiments has decreased, Figure 5.3 shows that there is still some variation *between* them. The models that used only feature groups O and T to predict churn have the lowest predictive accuracy of all models ($0.503 \leq \mu \leq 0.520$). Refer to Table 5.1 for the content of all feature groups.
The predictive accuracy of the model increased ($p \leq 0.05$), when O and T were combined with I. However, the predictive accuracy of the models predicting churn one or two months into the future was as good as the predictive accuracy of feature group I by itself



(a) All average AUC scores



(b) average AUC scores for models trained with a feature group combination that consists only of I, O and T



(c) average AUC scores for models trained with a feature group combination that consists only of I, O_r and T_r

Figure 5.3: Average AUC scores of 93 models. The samples were balanced in order to solve the problem of class imbalance.

(a simpler model). This is shown more clearly in Figure 5.3b, which contains the results of those models that used only static feature groups O and T, and I in their prediction.

Next to it is Figure 5.3c, which contains the results of those models that used only retrospective features O_r and T_r, and I. The models shown in this figure are some of the best performing ones. Compared to the static feature groups O and T, the feature group I was a relatively good feature to determine churn with. Compared to models that were trained with retrospective feature groups, however, models trained with only I perform relatively bad. More specifically: Any model that uses the feature group T_r will outperform a model that used only I.
However, Figure 5.3a contains even higher AUC scores. This means that even though the feature groups I, O an T were not very good predictors of churn by themselves, they can increase the predictive accuracy of other models.

The feature combination that was able to predict churn one month into the future most accurately, was the combination that contained feature groups I, O_r, T_r, T ($\mu = 0.666$, $\sigma = 0.046$). However, it did not predict churn in one month significantly better than the simpler models that used only feature groups I, T_r, T ($\mu = 0.651$, $\sigma = 0.048$, $p = 0.181$); I, O, T_r ($\mu = 0.647$, $\sigma = 0.047$, $p = 0.070$); I, O_r, T_r ($\mu = 0.661$, $\sigma = 0.046$, $p = 0.290$); or I, T_r ($\mu = 0.645$, $\sigma = 0.047$, $p = 0.072$). 'Simpler'

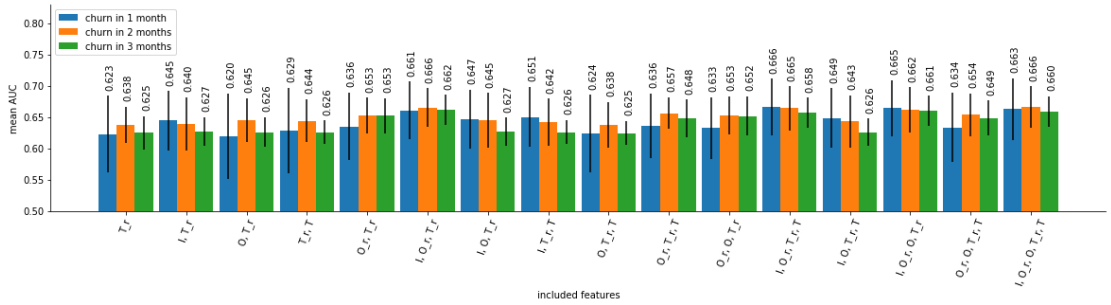


Figure 5.4: All models that contained at least the feature group T_r. They show only minor variations between them.

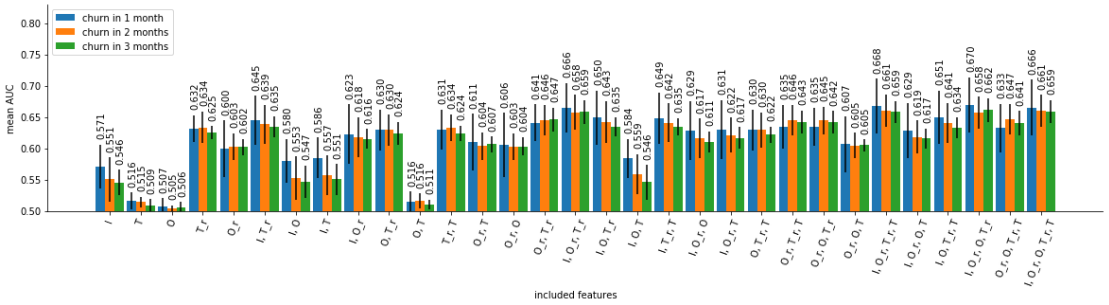
in this case is defined as 'using less features'. Note that feature group T contains more features than O, which makes O a 'simpler' group than T. However, T is not simpler than O, nor is it simpler than T_r as they contain exactly as many features.
The feature combination that was able to predict churn two months into the future most accurately, was the combination that contained all feature groups I, O_r, O, T_r and T. ($\mu = 0.666, \sigma = 0.033$). However, it did not predict churn in two months significantly better than the simpler models that used only feature groups O_r, O, T_r, T ($\mu = 0.654$, $\sigma = 0.034$, $p = 0.425$); I, O_r, T_r, T ($\mu = 0.665$, $\sigma = 0.035$, $p = 0.531$); O_r, O, T_r ($\mu = 0.653$, $\sigma = 0.031$, $p = 0.381$); O_r, T_r, T ($\mu = 0.657$, $\sigma = 0.026$, $p = 0.511$); I, O_r, T_r ($\mu = 0.666$, $\sigma = 0.031$, $p = 0.771$); O_r, T_r ($\mu = 0.653$, $\sigma = 0.029$, $p = 0.370$); or T_r, T ($\mu = 0.644$, $\sigma = 0.034$, $p = 0.093$).
The feature combination that was able to predict churn three months into the future most accurately, was the combination that contained feature groups I, O_r, and T_r. ($\mu = 0.662, \sigma = 0.024$). However, it did not predict churn in three months significantly better than the simpler model that used only feature groups O_r, T_r ($\mu = 0.653$, $\sigma = 0.028$, $p = 0.319$).

The one feature group that occurs in every optimal feature combination, as well as all simpler models that do not differ significantly in performance, is feature group T_r. Figure 5.4 shows that when T_r is used to predict churn, adding other feature groups to the model does not influence the model's performance much.

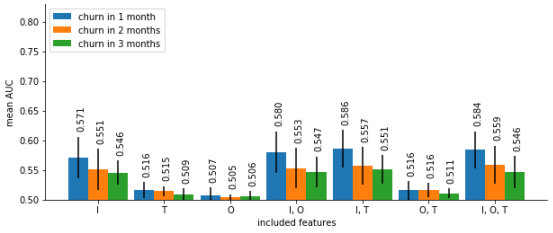
Testing on an unbalanced test set

In order to determine how well the model would perform in a realistic setting (i.e., on 'de Amazones', where unbalanced data is the norm), The experiments were also performed on an unbalanced test set.

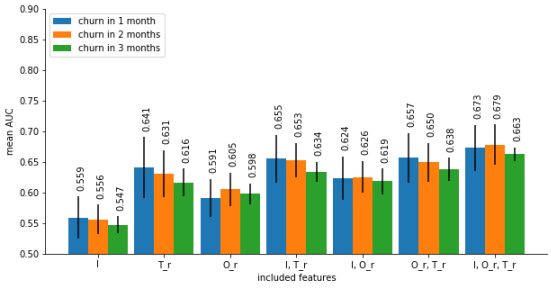
Figure 5.5a shows the 93 results of the conducted experiment, during which the models were trained on balanced data, but tested on unbalanced data. At a first glance, the



(a) All average AUC scores



(b) Average AUC scores for models trained with a feature group combination that consists only of I, O and T



(c) Average AUC scores for models trained with a feature group combination that consists only of I, O_r and T_r

Figure 5.5: Average AUC scores of 93 models. The models were trained on balanced data to handle the problem of class imbalance, then evaluated on an unbalanced set to determine their predictive accuracy in a real world situation.

results shown in Figure 5.3a look similar to those presented in Figure 5.3a, which were trained and tested on balanced data. In order to determine whether and where there are any differences in performance, the results will be more thoroughly examined.Recall that all means and standard deviations can be found in Appendix A, and that one can look up whether two models differ significantly in performance in Appendix B.

What was observed earlier for the models that were evaluated on a balanced test set, largely holds for the models that were trained on an unbalanced test set, as well. The models that use only O and T to predict churn are, like before, the worst performing models for all three dependent variables ($0.505 \leq \mu \leq 516$) . The predictive accuracy of the model increased ($p \leq 0.05$), when O and T were combined with I. Figure 5.5b shows how those models that use only O, T and I, compare.
Next to it is Figure 5.5c, which contains the results of those models that used only retrospective features O_r and T_r, and I. Like before, these models are amongst the best performing ones. The figure also shows that although feature group I was a relatively good predictor compared to the static feature groups, it is outperformed by any model that uses T_r to predict churn. However, Figure 5.3a contains even higher AUC scores.

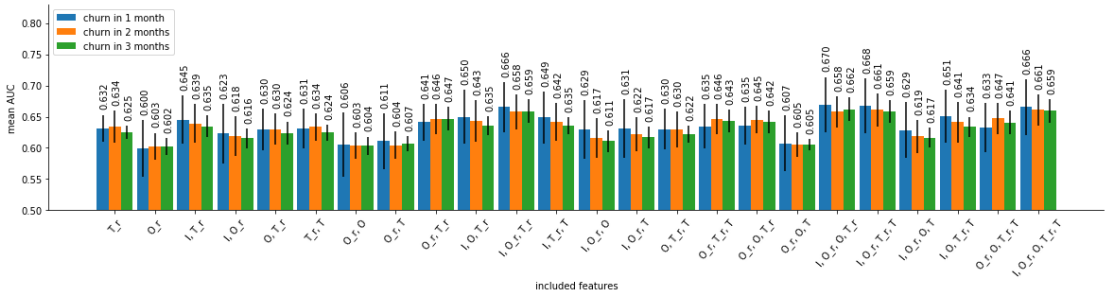


Figure 5.6: All models, evaluated on an unbalanced test set, that contained at least the feature group T_r or O_r.

This means that even though the feature groups I, O an T were not very good predictors of churn by themselves, they can increase the predictive accuracy of other models.

The feature combination that was able to predict churn one month into the future most accurately on unbalanced data, was the combination that contained feature groups I, O_r, O, and T_r ($\mu = 0.670$, $\sigma = 0.044$). However, it did not predict churn in one month significantly better than the simpler model that used only feature groups I, O_r, T_r ($\mu = 0.666$, $\sigma = 0.040$, $p = 0.122$).
The feature combination that was able to predict churn two months into the future most accurately on unbalanced data, was: I, O_r, T_r, and T. ($\mu = 0.661, \sigma = 0.026$). However, it did not predict churn in two months significantly better than the simpler models that used only feature groups O_r, O, T_r ($\mu = 0.645$, $\sigma = 0.022$, $p = 0.128$); O_r, T_r, T ($\mu = 0.646$, $\sigma = 0.024$, $p = 0.149$); I, O_r, T_r ($\mu = 0.658$, $\sigma = 0.028$, $p = 0.360$); or O_r, T_r ($\mu = 0.646$, $\sigma = 0.024$, $p = 0.165$)
Finally, the same feature group combination that was optimal for predicting churn in one month: I, O_r, O, T_r, was optimal to predict churn three months into the future ($\mu = 0.662$, $\sigma = 0.019$). However, it did not predict churn in three months significantly better than the simpler model that used only feature groups I, O_r, T_r ($\mu = 0.659$, $\sigma = 0.019$, $p = 0.093$).

As was the case when the models were evaluated on the balanced data, the feature group T_r was used by every model that predicted churn most accurately, as well as in all simpler models that did not differ significantly in predictive accuracy. In addition, O_r was used in every one of these models as well. Figure 5.6 shows all models that use at least T_r or O_r to predict churn.

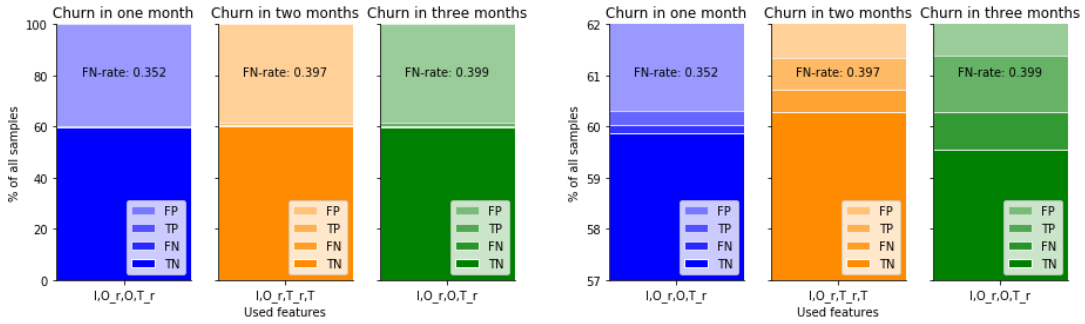
Table 5.2 summarises the findings presented in this section. It shows which models were the best predictors of churn, and which simpler models (models that used less features) demonstrated a comparable predictive accuracy.

Finally, in order to determine the applicability of these models in a realistic situ-

Table 5.2: All models that predicted churn most accurately, as measured with the AUC. The rightmost colum lists all simpler models (those that use less features) that had a similar AUC score. The simplest model in each category is indicated with a star.

Test set	Churn in ... month(s)	Best model	Comparable, simpler models	
Balanced	one	(I, O_r, T_r, T)	(I, T_r, T)	
			(I, O, T_r)	
			(I, O_r, T_r)	
			(I, T_r)	*
	two	(I, O_r, O, T_r, T)	(O_r, O, T_r, T)	
			(I, O_r, T_r, T)	
			(O_r, O, T_r)	
			(O_r, T_r, T)	
			(I, O_r, T_r)	
			(O_r, T_r)	
			(T_r, T)	*
	three	(I, O_r, O, T_r)	(O_r, T_r)	*
Unbalanced	one	(I, O_r, O, T_r)	(I, O_r, T_r)	*
	two	(I, O_r, T_r, T)	(O_r, O, T_r)	
			(O_r, T_r, T)	
			(I, O_r, T_r)	
			(O_r, T_r)	*
	three	(I, O_r, O, T_r)	(I, O_r, T_r)	*

ation, a look was taken at the false negative rate of each of them. Figure 5.7 shows the distribution of False Positives (FP), True Positives (TP), False Negatives (FN) and True Negatives (TN) in the predictions of the three models that predicted churn most accurately in one, two and three months. The figure clearly shows that the number of FP and TN is very high, compared to the number of TP and FN. Explicitly: the ratio FN+TP:TN+FP is equal to the ratio positive:negative samples in the test set. As the test set was unbalanced, it is thus no surprise that there are many more TN and FP than FN and TP. To calculate the FN-rate, only FN and TP are needed. The false negative rate is the number of wrongly predicted positive samples (FN), divided by the total number of positive samples (FN+TP). In order to be able to take a better look at how TP and FN compare, Figure 5.7b shows percentiles 57-62 from Figure 5.7a. From this figure, it becomes clear that all three models predicted the churn of churning patients more often correctly than not: Out of all times that a user was about to churn within one month, 64.81% was predicted as such ($FN = 0.352$). Out of all times that a user was about to churn within two months, 60.27% was predicted as such ($FN = 0.397$). And out of all times that a user was about to churn within three months, 60.09% was predicted as such ($FN = 0.399$).



(a) The distribution of FP, TP, FN and TN. Figure 5.7b shows a close-up of the percentiles 57-62.

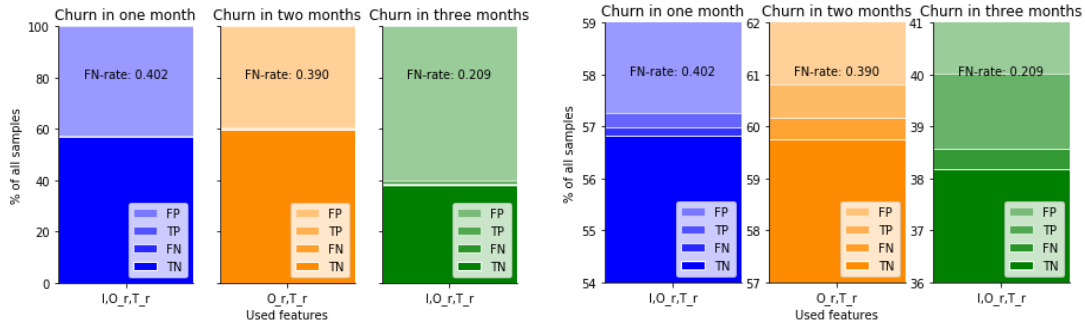
(b) A detail of Figure 5.7a, showing the distribution of TP and FN more clearly.

Figure 5.7: The percentage of samples that was FP, TP, FN, and TN, by the models with the highest predictive accuracy. The FN-rate of every model is written within each bar.

Figure 5.8a shows the distribution of FP, TP, TN and FN for the three *simplest* models that had a similar ROC-AUC score. These models were indicated with an asterisk in Table 5.2. The figure shows that the number of false positives is much higher when churn in three months was predicted with this 'simpler' classifier, compared to when it was predicted with the best. In order to be able to compare the TP and FN as well, Figure 5.8b shows the percentiles around TP and FN: 54-59, 57-62, and 36-41, respectively. Out of all times that a user was about to churn within one month, 59.81% was predicted as such ($FN = 0.402$). Out of all times that a user was about to turn within two months, 60.98% was predicted as such ($FN = 0.390$). Out of all times that a user was about to churn within three months, 79.10% was predicted as such ($FN = 0.209$).

To get a better idea of how good this performance is, these results are compared to a baseline. When churn data is considered, it is common to determine a baseline by making the same prediction for all samples, usually the minority class. This reflects the number of correctly classified users when they are all assumed not to churn. The AUC in this case is about 0.5, but the FNR is exactly 1 (assuming that the positive class is the minority class): as the positive class was never predicted, FN = 0, which makes the formula for the FNR equal to $\frac{FN}{FN}$, which equals 1.
Another common way to determine the baseline of a model is to predict scores randomly. In this case, too, the AUC is about 0.5. The FNR in this case is about 0.5 as well. Any model that has a higher AUC and lower FNR performs better than these baselines ('default' and 'random', respectively).

The best model for churn in one month, (I, O_r, O, T_r), had an ROC-AUC score of 0.670 ($\sigma = 0.044$) and a FNR of 0.352. The best model for churn in two months, (I, O_r, T_r, T), had an ROC-AUC score of 0.661 ($\sigma = 0.026$) and a FNR of 0.397. The



(a) The distribution of FP, TP, FN and TN. Figure 5.7b shows a close-up of the percentiles around the TP and FN.

(b) A detail of Figure 5.7a, showing the distribution of TP and FN more clearly.

Figure 5.8: The percentage of samples that was FP, TP, FN, and TN, by the models with the highest predictive accuracy. The FN-rate of every model is written within each bar.

best model for churn in three months, (I, O_r, O, T_r), had an ROC-AUC score of 0.662 ($\sigma = 0.019$) and a FNR of 0.399. These models thus performed better than 'default', and better than 'random'.

The simplest model that did not differ significantly in predictive accuracy from the best model for churn in one month (I, O_r, T_r), had an ROC-AUC score of 0.66 ($\sigma = 0.040$) and a FNR of 0.402. The model for churn in two months (O_r, T_r), had an ROC-AUC score of 0.646 ($\sigma = 0.024$) and a FNR of 0.390. The model for churn in three months (I, O_r, T_r), had an ROC-AUC score of 0.659 ($\sigma = 0.019$) and a FNR of 0.209. These models, too, performed better than default and random predictors.
Table 5.3 summarises these results.

Table 5.3: The AUC and FNR scores of the best models tested on unbalanced data, and the simplest model that did not perform significantly worse, compared to those. All models perform better than the 'default' ($AUC \geq 0.5$, $FNR \leq 1$) and better than 'random' ($AUC \geq 0.5$, $FNR \leq 0.5$).

model	score	churn in one month	churn in two months	churn in three months
'best'	AUC	0.670 ± 0.044	0.661 ± 0.026	0.662 ± 0.019
	FNR	0.352	0.397	0.399
'simplest'	AUC	0.646 ± 0.040	0.646 ± 0.024	0.659 ± 0.019
	FNR	0.402	0.390	0.209

Chapter 6

Discussion

This thesis addressed the issue of churn in online communities. More specifically, it aimed to construct a model that was able to predict whether cancer patients that participate on 'de Amazones' will churn within the next three months. The features that were used to train this model, were all measured from the users' forum posts.
Three research questions were formulated to guide the research. These questions are repeated below:

1. How accurately can churn be predicted from a given point in time, one, two, and three months into the future?

2. Which features, measurable from a user's forum posts, are most important for the prediction of churn?

3. How can the problem of class imbalance best be handled?

To answer these questions, numerous models were trained, tested and compared.

Models were constructed on data that was weighted. Because the runtime of the program would otherwise be very long, a subsample of 5% was used to run this experiment on. The results showed that there were large variations in the predictive accuracy of a model between the ten folds. An effort to solve this problem was done: the experiment was also run with five folds, instead of ten. comparing the standard deviations of the models' AUC scores revealed that this did indeed improve the predictive performance of the models. However, the standard deviations were still large, which made any interpretation of them unreliable.

It seems likely that the results would improve noticeably, if more data would be used. Because the original ratio positive:negative samples remained in the subsampled data that was used for these experiments, there were only very few positive samples available in the train set of every fold ($0.6 * \frac{1}{k} * |s^1|$, where $|s^1|$ was 125, 320, or 545 for churn in one, two and three months respectively). These were, despite the extra penalty that

Table 6.1: The number of models that each feature group occurred in, sorted by frequency

feature group	balanced test	unbalanced test
T_r	15	9
O_r	11	9
I	9	7
T	7	2
O	5	3
total	15	9

was put on the wrongly classified samples from the minority class, probably often too few for the algorithm to learn from accurately. This hypothesis was supported by the increase in performance when 5 folds were used instead of 10, which lead to larger fold sizes, with a higher absolute number of positive samples.

The second way in which the problem of class imbalance was handled, was balancing. Because the data was already subsampled inexplicitly, as part of the balancing method, the run time of this experiment was already very manageable. The experiments contained 5034 (for churn in one month), 12810 (for churn in two months) or 21952 (for churn in three months) samples, of which half was positive (of which 60% was used to train the model on). The results showed that there were only small variations in the predictive accuracy of a model between the ten folds, which made interpretation more reliable. The predictive accuracy of the models was comparable when the models that were trained on balanced data were tested on unbalanced data.

However, when the models that returned the highest predictive accuracy were compared to the models that had a comparable predictive accuracy but were simpler, one notices that all of these models contain at least the feature group T_r. In addition, many contain feature group O_r. Feature group I occurs in a little over half of the models, and feature groups O and T are in only some of the models. This comparison reveals that the most important feature groups to predict churn with, are T_r and O_r. Table 6.1 summarises the results, which were earlier presented (in a different format) in Table 5.2.

These results confirm the hypothesis that the retrospective features have a higher predictive accuracy than static features.
However, the possibility of data leakage must also be considered. As YellowRoad (2017) points out, "if by eliminating just a single feature from the input set you experience a dramatic reduction in performance, there is a high chance that the eliminated feature contains leakage." This may for example be the case if the eliminated feature is a *proxy feature*. A proxy feature is a feature that is so strongly correlated with the dependent variable that is can almost be used as the dependent variable. In this thesis, it was observed that the features O_r and T_r had a very big influence on the predictive accuracy.

Let us consider the feature groups O_r and T_r in more detail, and determine to what extent there might be data leakage here. Both variables describe the past 30 days of a single user, as seen from a specific point in time. Bins that a forum user has not been active in will have the value '0'. If a user has been largely inactive in the month prior to a specific sample, the retrospective feature values will be (close to) zero. This number does not just reflect the average value of the static counterpart of that feature, but in a way also the activity level of a user. The closer the retrospective feature value is to zero, the less active the user has been. Perhaps this inactivity is not a sign of the churn that will come, but a sign that churn has already begun. To find out what the exact role and influence of these retrospective features is, additional research is needed that delves deeper into the dependence and independence of the measured features on each other.

The results also show that the textual features outperform the opinion features. Further explorations are necessary to determine why exactly this is the case. One possible explanation is that users show different kinds of activity over time, which are reflected in the textual features. A user may for example have less questions, or write shorter posts as time passes. Eventually, this leads to churn.
Another possible explanation is that feature groups T and T_r consist of three features (post length, sentence length and question ratio), while feature group O and O_r consist of two (sentiment and subjectivity). The feature groups T_r and T therefore contain more information than the feature groups O_r and O do.
The performance of both the static and retrospective opinion variables may also increase when the sentiment miner is elaborated such that nouns and verbs can also be tagged with sentiment and subjectivity values. In Chapter 3, an example was given of a sentence that was tagged with a sentiment and subjectivity value of both 0. The example was 'ik hou niet van taart' (*I do not like pies*). Although one might argue that there is indeed little sentimental value in this sentence, there definitely is some degree of subjectivity: the sentence is phrased in the first person singular ('Ik', *I*), after which a verb expressing an emotional state ('houden van', *to love*) follows. However, this subjectivity was not measured, because Pattern only looks at the adjectives in a sentence.

To get a better idea of how these models would perform in a realistic setting, the false negative rate (FNR) of the best models, and the simplest comparable models was determined. The FNR reflects the number of churning users that was not identified as a churner. This rate is important, because by targeting the potential churners, the churn rate can be decreased. The results showed that the FNR of the best models increased slightly as the churn date got closer. Users who churned within one month were identified correctly 64.1% of the time, users who churned within two months were identified correctly 60.27% of the time, and users who churned within three months were identified correctly 60.09% of the time. Interestingly, the FNR of the simpler, comparable models decreased as the churn date got closer. Users who churned within one month were identified correctly 59.81% of the time, users who churned within two months were identified correctly 60.98% of the time, and users who churned within three months were identified

correctly 79.10% of the time.
For predicting churn in one month, the simpler model performed worse than the model with the highest AUC. For churn in two months, the models were comparable. For churn in three months, the simpler model performed better than the model with the highest AUC, and actually had the lowest FNR of all.

It is in line with intuitions that it is easier to predict whether someone will churn in the near future than it is to predict churn in a less near future. However, when churn in one month is predicted, less samples are available than when churn in three months is predicted. This may make it harder to learn sufficiently enough about the behaviour of a churner.
The results seem to indicate that when the models are used with the highest predictive accuracy, these two contrasting factors middle out. The FNR of all three best models is about the same. However, when simpler models are used, the results seem to indicate that the imbalance of the data is a much more decisive factor: those who churned within three months were identified correctly more often than those who churned within one month. It seems likely that the FNR of all models will increase when more data is used. Especially when more positive samples are used.

This hypothesis is supported by the results from the explorations that were done to determine which parameters should be passed to grid search in the experiments. These explorations were done with 5-fold cross validation, which resulted in much larger train sets to develop the parameters on. With the same parameter options that were used in the experiments, the explorations showed that it is possible to obtain an AUC score of 0.851 ± 0.006. As the only difference between this exploration and the experiment (which obtained an AUC score of 0.660 ± 0.024) is the amount of data that was used, it seems undeniable that using more data will lead to better results.
There is one critical point which needs to be considered here. The explorations on which parameters should be used during the experiments were done on the same exact dataset as the experiments were performed. This is a form of data leakage, even if there was still a range of parameter settings passed to grid search in the experiments. A follow-up should redo the exploration, but on a completely separate set of the data. Another option would be to do the exploration on the train and development sets of the data, although this approach is difficult because these are split into k parts: one for every k^{th} fold.

The current approach, with ten separate folds, was chosen because it allowed to compare the behaviour of the models on balanced and unbalanced test sets, while minimising the chance of leakage from the train into the test set. The comparison between them revealed that they behaved comparably when the models were trained on balanced data. A follow-up of this study could use a different approach, as the current proved that the results were similar. Another approach which allows for a bigger training set (For example k-fold cross validation) is expected to increase the predictive accuracy of

the models. In addition, the effect of the parameter scale_pos_weight could be explored, even when the train data is balanced. Perhaps it allows the algorithm to focus more on the positive samples, which increases their predictive accuracy.

The results of this study were finally compared to a baseline. This comparison revealed that the best models, as well as the simpler, comparable ones, performed better than a predictor which would always predict the negative class (the majority class). The comparison also revealed that the algorithm performed better than chance.

Chapter 7

Conclusion

This thesis aimed to answer the question **how accurately churn can be predicted**, and **which features were the most important factors in this prediction**. In addition, **the class imbalance problem** was addressed.

Models were trained on balanced data, and later tested on unbalanced data in order to deterimine the practical implications of the models. The results from these experiments showed that there was one 'best model' for each of three dependent variables, and several 'simpler, but comparable' models (in terms of number of features they contain). Table 5.2 gave an overview of these. Every 'best model' and 'simplest, comparable' model was further examined with the false negative rate. This revealed that of the three dependent variables, churn in three months could be most accurately predicted when it was evaluated with the False negative rate. This was somewhat counter to intuitions, but can likely be explained by a lack of samples in the data set.

Additional comparisons between all 'best' and 'simpler' models revealed that the feature groups O_r and T_r were the most important predictors of churn. These were the two retrospective feature groups, and were already a priori expected to cause a better performance of a model. In Chapter 6, the possibility that these features are proxy features was examined. More research is needed to determine whether this is the case.

These results are good news for the Dutch breast cancer association (BVN), and similar organisations that host a patient forum. The results indicate that it is possible to use fairly simple and easy-to-annotate features that can be annotated on any forum, to predict which users will churn. Although some additional research would still be needed, these results are very promising. Moreover, implementing such a predictive algorithm with for example BVN would not solely benefit the forum. Patients are also (indirectly) affected, as the forum becomes more attractive once it is more active. And because an active forum is more engaging than a dormant one, many old and new users will be benefited with the positive effects of forum participation. They will become better informed, and feel more empowered in general. The forum will once again be a very

powerful tool to enable patients to have the best quality of life.

Appendix A

Result tables

In this appendix, the average AUC of all different models trained and tested on balanced data is shown. The appendix also contians the average AUC of all different models trained on balanced, but tested on unbalanced data. The values in the table are probably easiest interpreted next to Figure 5.3a and Figure 5.5a, which visualise the predictive accuracy of each of the 93 models that was tested on balanced and unbalanced data, respectively. Whether a difference between two models (that predict the same dependent variable) is observed to be significant can be looked up in Appendix A.

Table A.1: Average AUC scores of models predicting churn one, two, and three months into the future, using different combinations of feature groups. Models were evaluated on balanced data.

Included features	Churn in 1 month	Churn in 2 months	Churn in 3 months
I	0.560 ± 0.035	0.557 ± 0.048	0.535 ± 0.026
T	0.520 ± 0.020	0.519 ± 0.017	0.509 ± 0.013
O	0.504 ± 0.017	0.503 ± 0.012	0.508 ± 0.014
T_r	0.623 ± 0.062	0.638 ± 0.029	0.625 ± 0.027
O_r	0.597 ± 0.071	0.602 ± 0.021	0.606 ± 0.025
I, T_r	0.645 ± 0.047	0.640 ± 0.042	0.627 ± 0.023
I, O	0.576 ± 0.037	0.562 ± 0.046	0.538 ± 0.031
I, T	0.580 ± 0.034	0.567 ± 0.042	0.537 ± 0.032
I, O_r	0.612 ± 0.059	0.624 ± 0.039	0.611 ± 0.025
O, T_r	0.620 ± 0.069	0.645 ± 0.035	0.626 ± 0.024
O, T	0.513 ± 0.020	0.511 ± 0.023	0.512 ± 0.008
T_r, T	0.629 ± 0.068	0.644 ± 0.034	0.626 ± 0.019
O_r, T	0.607 ± 0.074	0.602 ± 0.028	0.610 ± 0.028
O_r, O	0.604 ± 0.068	0.603 ± 0.027	0.609 ± 0.029
O_r, T_r	0.636 ± 0.054	0.653 ± 0.029	0.653 ± 0.028
I, O_r, T_r	0.661 ± 0.046	0.666 ± 0.031	**0.662 ± 0.024**
I, O, T_r	0.647 ± 0.047	0.645 ± 0.044	0.627 ± 0.023

Table A.1: continuted

Included features	Churn in 1 month	Churn in 2 months	Churn in 3 months
I, O, T	0.574 ± 0.030	0.568 ± 0.044	0.537 ± 0.034
I, T_r, T	0.651 ± 0.048	0.642 ± 0.038	0.626 ± 0.020
I, O_r, O	0.613 ± 0.059	0.621 ± 0.040	0.606 ± 0.027
I, O_r, T	0.624 ± 0.055	0.625 ± 0.042	0.612 ± 0.029
O, T_r, T	0.624 ± 0.062	0.638 ± 0.036	0.625 ± 0.020
O_r, T_r, T	0.636 ± 0.052	0.657 ± 0.026	0.648 ± 0.030
O_r, O, T_r	0.633 ± 0.049	0.653 ± 0.031	0.652 ± 0.032
O_r, O, T	0.605 ± 0.070	0.601 ± 0.028	0.609 ± 0.028
I, O_r, T_r, T	**0.666 ± 0.046**	0.665 ± 0.035	0.658 ± 0.024
I, O_r, O, T	0.622 ± 0.054	0.625 ± 0.038	0.612 ± 0.030
I, O, T_r, T	0.649 ± 0.048	0.643 ± 0.041	0.626 ± 0.021
I, O_r, O, T_r	0.665 ± 0.045	0.662 ± 0.036	0.661 ± 0.024
O_r, O, T_r, T	0.634 ± 0.056	0.654 ± 0.034	0.649 ± 0.028
I, O_r, O, T_r, T	0.663 ± 0.050	**0.666 ± 0.033**	0.660 ± 0.024

Table A.2: Average AUC scores of models predicting churn one, two, and three months into the future, using different combinations of feature groups. Models were evaluated on unbalanced data.

Included features	Churn in 1 month	Churn in 2 months	Churn in 3 months
I	0.571 ± 0.035	0.551 ± 0.035	0.546 ± 0.020
T	0.516 ± 0.014	0.515 ± 0.008	0.509 ± 0.009
O	0.507 ± 0.014	0.505 ± 0.005	0.506 ± 0.009
T_r	0.632 ± 0.021	0.634 ± 0.025	0.625 ± 0.011
O_r	0.600 ± 0.046	0.603 ± 0.021	0.602 ± 0.013
I, T_r	0.645 ± 0.039	0.639 ± 0.031	0.635 ± 0.017
I, O	0.580 ± 0.035	0.553 ± 0.034	0.547 ± 0.026
I, T	0.586 ± 0.032	0.557 ± 0.032	0.551 ± 0.025
I, O_r	0.623 ± 0.047	0.618 ± 0.032	0.616 ± 0.016
O, T_r	0.630 ± 0.033	0.630 ± 0.025	0.624 ± 0.018
O, T	0.516 ± 0.015	0.516 ± 0.012	0.511 ± 0.008
T_r, T	0.631 ± 0.032	0.634 ± 0.022	0.624 ± 0.012
O_r, T	0.611 ± 0.045	0.604 ± 0.022	0.607 ± 0.012
O_r, O	0.606 ± 0.051	0.603 ± 0.021	0.604 ± 0.015
O_r, T_r	0.641 ± 0.030	0.646 ± 0.024	0.647 ± 0.019
I, O_r, T_r	0.666 ± 0.040	0.658 ± 0.028	0.659 ± 0.019

Table A.2: continuted

Included features	Churn in 1 month	Churn in 2 months	Churn in 3 months
I, O, T_r	0.650 ± 0.043	0.643 ± 0.033	0.635 ± 0.015
I, O, T	0.584 ± 0.031	0.559 ± 0.032	0.546 ± 0.027
I, T_r, T	0.649 ± 0.041	0.642 ± 0.031	0.635 ± 0.014
I, O_r, O	0.629 ± 0.047	0.617 ± 0.032	0.611 ± 0.017
I, O_r, T	0.631 ± 0.047	0.622 ± 0.028	0.617 ± 0.017
O, T_r, T	0.630 ± 0.033	0.630 ± 0.029	0.622 ± 0.013
O_r, T_r, T	0.635 ± 0.035	0.646 ± 0.024	0.643 ± 0.018
O_r, O, T_r	0.635 ± 0.030	0.645 ± 0.022	0.642 ± 0.018
O_r, O, T	0.607 ± 0.045	0.605 ± 0.020	0.605 ± 0.010
I, O_r, T_r, T	0.668 ± 0.044	**0.661 ± 0.026**	0.659 ± 0.018
I, O_r, O, T	0.629 ± 0.044	0.619 ± 0.027	0.617 ± 0.016
I, O, T_r, T	0.651 ± 0.043	0.641 ± 0.032	0.634 ± 0.016
I, O_r, O, T_r	**0.670 ± 0.044**	0.658 ± 0.025	**0.662 ± 0.019**
O_r, O, T_r, T	0.633 ± 0.039	0.647 ± 0.025	0.641 ± 0.019
I, O_r, O, T_r, T	0.666 ± 0.045	0.661 ± 0.026	0.659 ± 0.019

Appendix B

Significance matrices

In this appendix, the significance of differences between models that predicted the same dependent variable is presented. This significance is presented in a confusion matrix. These results are best interpreted next to the result tables in Appendix B, or next to the Figures in Section 5.2. The first three matrices correspond to the models that were tested on the balanced datam the second three correspond to the models that were tested on unbalanced data. The color in each cell visualises the significance of the difference: darker colors indicate a higher significance. The figures start on the next page, due to their size.

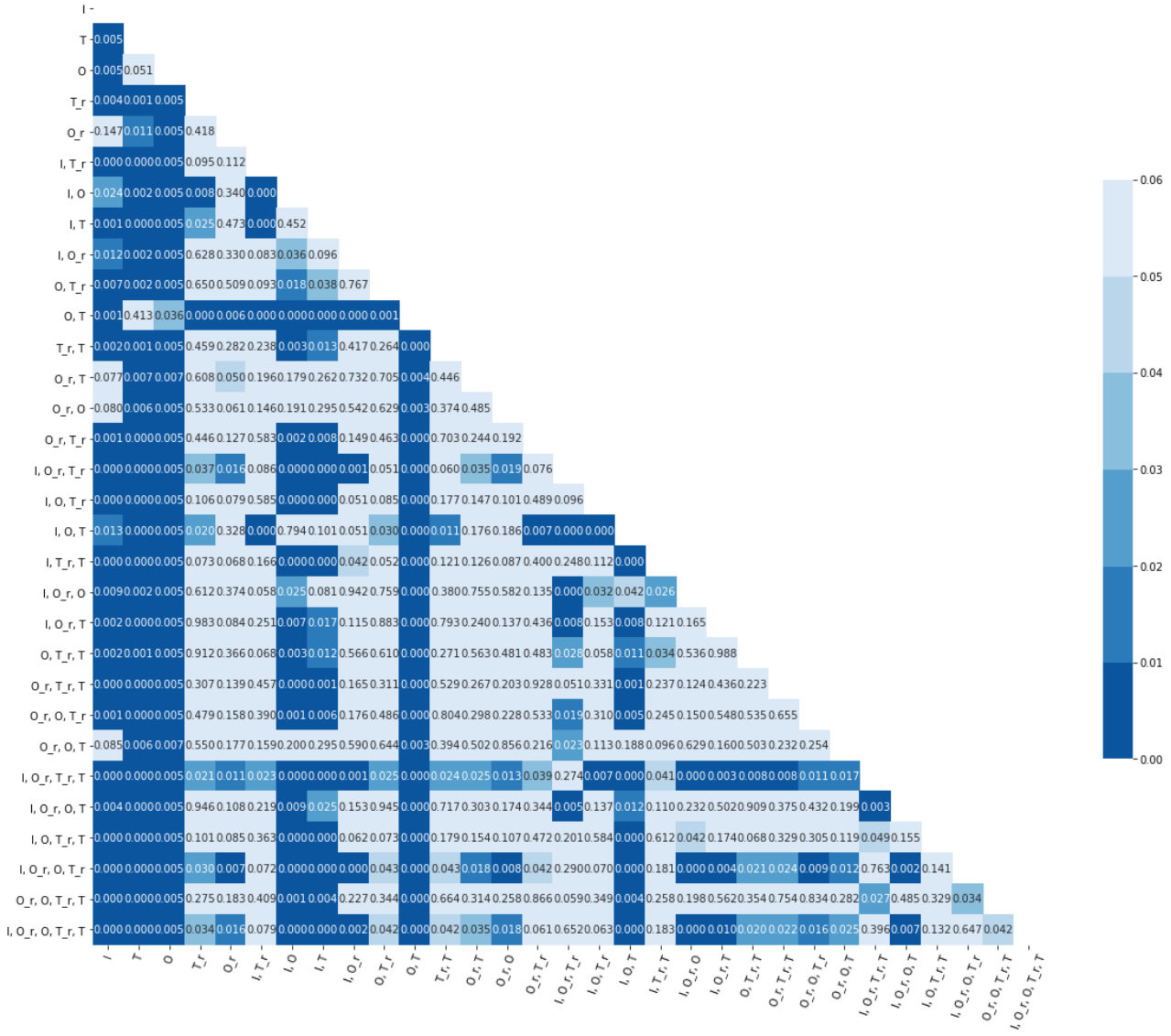


Figure B.1: The significance matrix for churn in one month, predicted on balanced data.

Figure B.2: The significance matrix for churn in two months, predicted on balanced data.

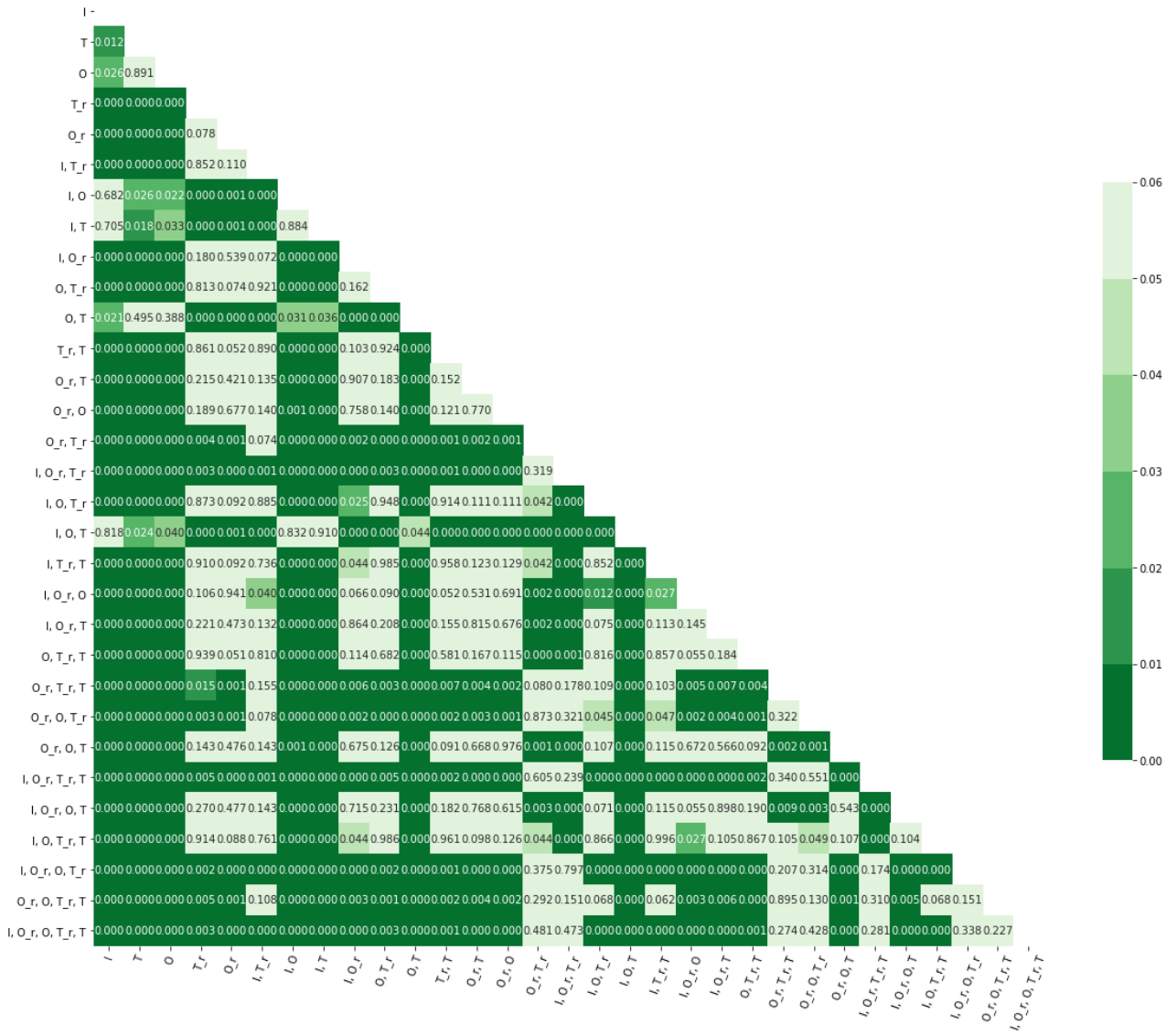


Figure B.3: The significance matrix for churn in three months, predicted on balanced data.

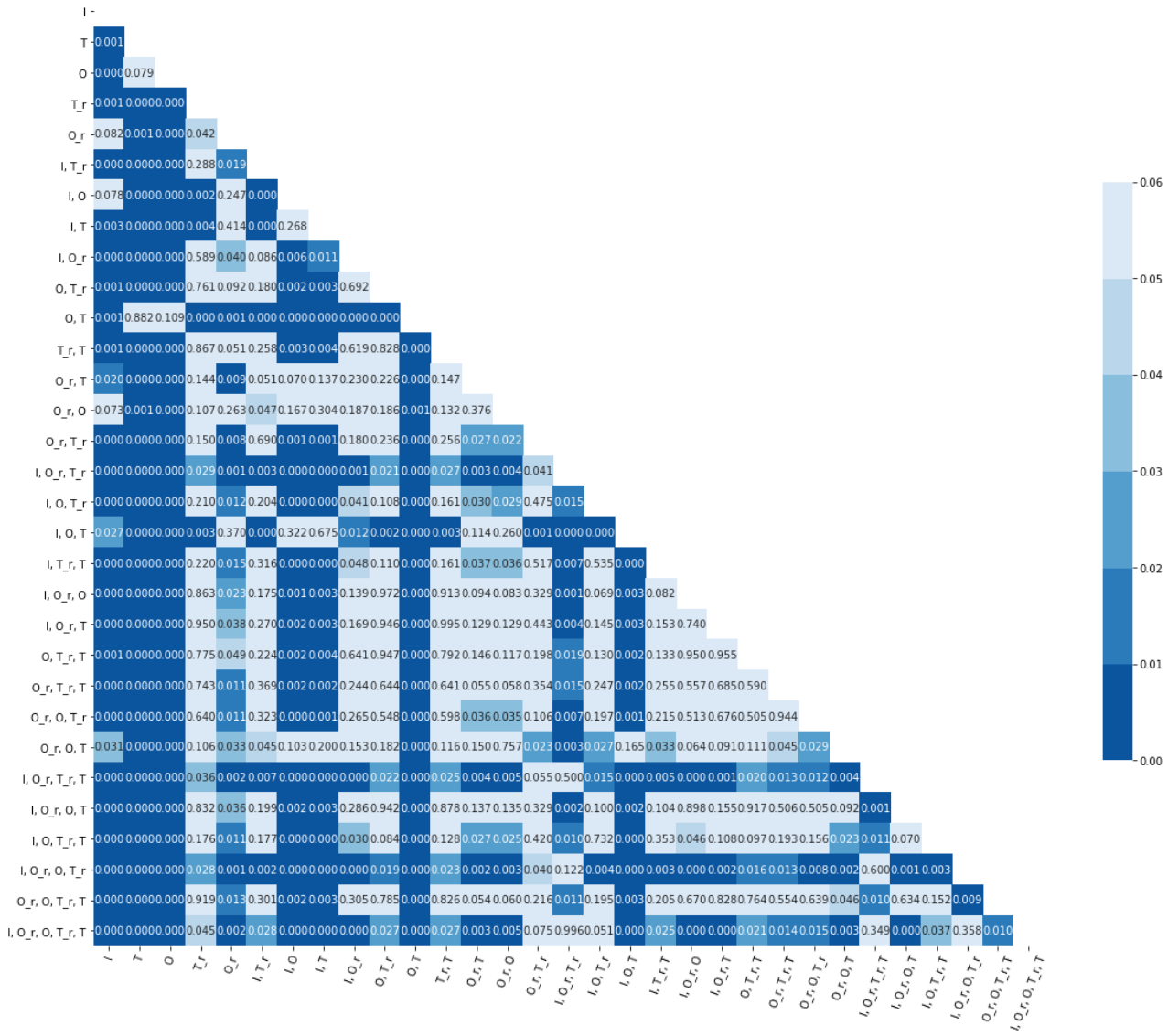


Figure B.4: The significance matrix for churn in one month, predicted on unbalanced data.

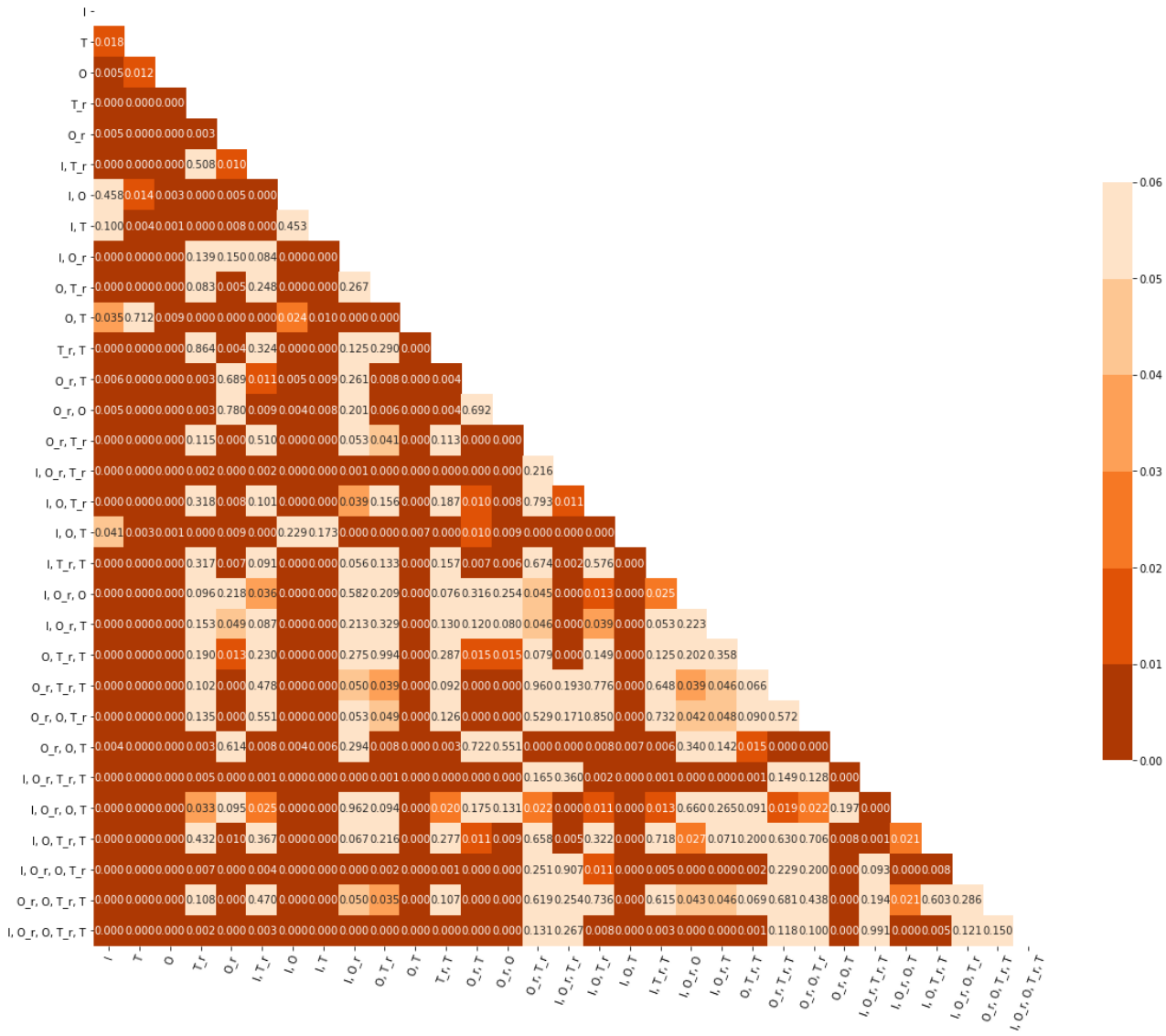


Figure B.5: The significance matrix for churn in two months, predicted on unbalanced data.

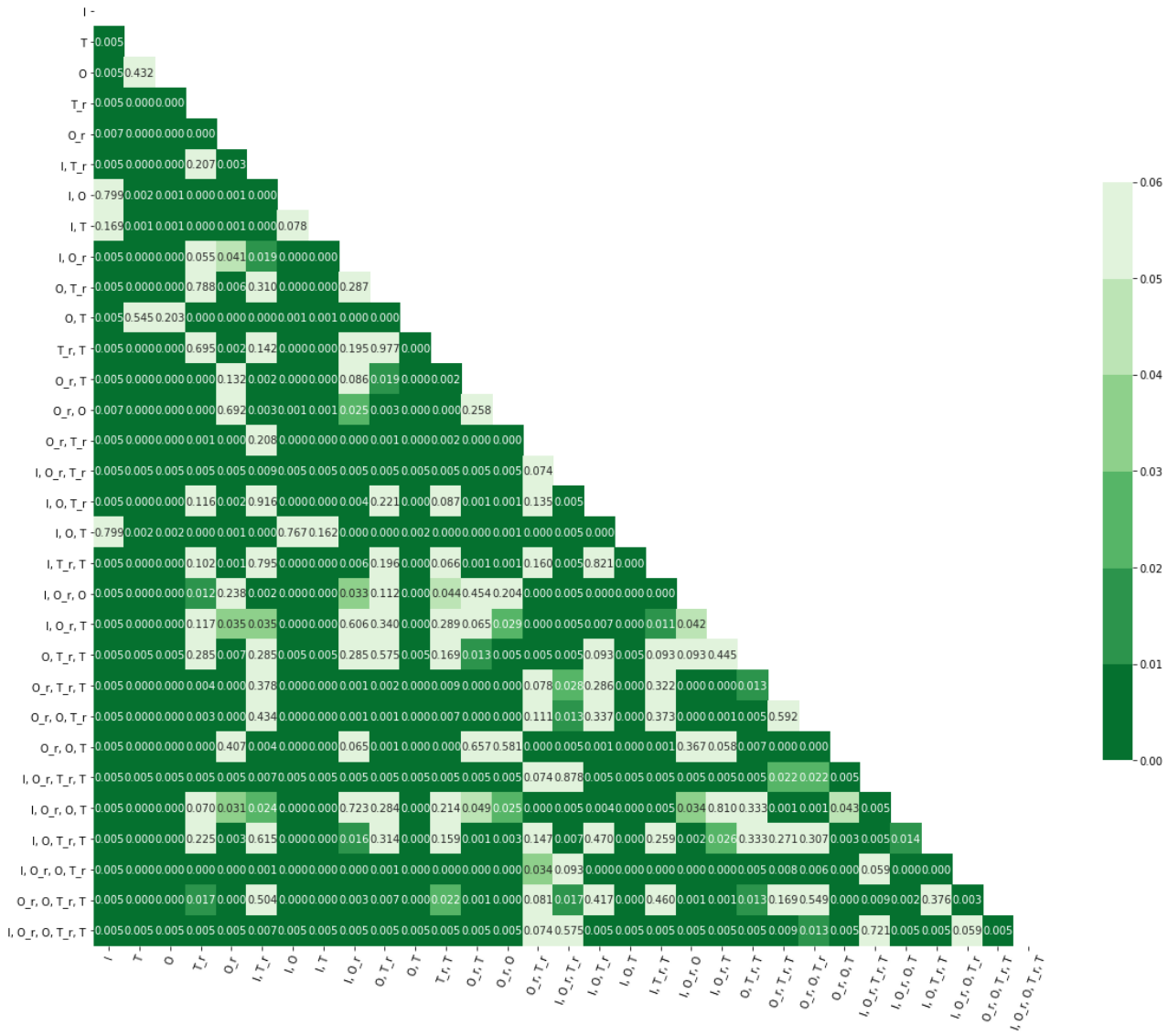


Figure B.6: The significance matrix for churn in three months, predicted on unbalanced data.

Bibliography

Ahn, J. H., Han, S. P., and Lee, Y. S. (2006). Customer churn analysis: Churn determinants and mediation effects of partial defection in the Korean mobile telecommunications service industry. *Telecommunications Policy*, 30(10-11):552–568.

Barone, M., Miniard, P., and Romeo, J. (2000). The Influence of Positive Mood on Brand Extension Evaluations. *Journal of Consumer Research*, 26(4):386–400.

Batenburg, A. E. (2015). *Being On-line Together: The Effects of Peer-led Online Support Community Participation on Breast Cancer Patients' Psychological Well-being*. PhD thesis, Vrije Universiteit Amsterdam.

Bo Pang, L. L. (2008). Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1):1–135.

Burez, J. and Van den Poel, D. (2009). Handling class imbalance in customer churn prediction. *Expert Systems with Applications*, 36(3 PART 1):4626–4636.

Casidy, R., Wymer, W., and O'Cass, A. (2018). Enhancing hotel brand performance through fostering brand relationship orientation in the minds of consumers. *Tourism Management*, 66:72–84.

Chen, T. and Guestrin, C. (2016). XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pages 785–794.

Cho, H. (2017). Speeding up gradient boosting for training and prediction.

De Smedt, T. and Daelemans, W. (2012a). Pattern for Python. *Journal of Machine Learning Research*, 13:2063–2067.

De Smedt, T. and Daelemans, W. (2012b). "Vreselijk mooi!" (terribly beautiful): A Subjectivity Lexicon for Dutch Adjectives. *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 3568–3572.

Distributed (Deep) Machine Learning Community (2015). XGBoost: Scalable and Flexible Gradient Boosting.

Friedman, J. H. (2001). Greedy Function Approximation : A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232.

Gallo, A. (2014). The Value of Keeping the Right Customers. *Harvard Business Review*.

Healey, B. and Hoek, J. (2014). Posting Behaviour Patterns in an Online Smoking Cessation Social Network: Implications for Intervention Design and Development. *PLoS ONE*, 9(9).

Karnstedt, M., Hennessy, T., Chan, J., Basuchowdhuri, P., Hayes, C., and Strufe, T. (2010). Churn in Social Networks. In Fuhrt, B., editor, *Handbook of Social Network Technologies and Applications*, chapter 9, pages 185–220. Springer Verlag.

Karnstedt, M., Rowe, M., Chan, J., Alani, H., and Hayes, C. (2011). The effect of user features on churn in social networks. *Proceedings of the 3rd International Web Science Conference on - WebSci '11*, pages 1–8.

Kawale, J., Pal, A., and Srivastava, J. (2009). Churn Prediction in MMORPGs: A Social Influence Based Approach. In *2009 International Conference on Computational Science and Engineering*, pages 423–428. IEEE.

Lauzier, S., Campbell, H. S., Livingston, P. M., and Maunsell, E. (2014). Indicators for evaluating cancer organizations' support services: Performance and associations with empowerment. *Cancer*, 120(20):3219–3227.

Lehmann, J., Lalmas, M., Yom-Tov, E., and Dupret, G. (2012). Models of User Engagement. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 164–175.

Loper, E. and Bird, S. (2002). NLTK: The Natural Language Toolkit. pages 63–70.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Richter, Y., Yom-Tov, E., and Slonim, N. (2010). Predicting customer churn in mobile networks through analysis of social groups. *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM 2010)*, pages 732–741.

Rust, R. T. and Zahorik, A. J. (1993). Customer satisfaction, customer retention, and market share. *Journal of Retailing*, 69(2):193–215.

Solc, T. (2018). Unidecode.

van Oortmerssen, G. (2016). Het Patiënten Forum Miner Project. *Leven met GIST*, 2(2):4–7.

van Oortmerssen, G., Raaijmakers, S., Sappelli, M., Verberne, S., Walasek, N., and Kraaij, W. (2017). Analyzing cancer forum discussions with text mining. *Proceedings of Second International Workshop on Extraction and Processing of Rich Semantics from Medical Texts*.

van Rossum, G. (1995). Python Tutorial. *Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam*.

van Uden-Kraan, C. F., Drossaert, C. H., Taal, E., Seydel, E. R., and van de Laar, M. A. (2008). Self-reported differences in empowerment between lurkers and posters in online patient support groups. *Journal of medical Internet research*, 10(2):1–9.

van Uden-Kraan, C. F., Drossaert, C. H. C., Taal, E., Lebrun, C. E. I., Drossaers-Bakker, K. W., Smit, W. M., Seydel, E. R., and van de Laar, M. A. (2007). Coping with somatic illnesses in online support groups: Do the feared disadvantages actually occur? *Computers in Human Behavior*, 24(2):309–324.

van Uden-Kraan, C. F., Drossaert, C. H. C., Taal, E., Seydel, E. R., and van de Laar, M. A. (2009). Participation in online patient support groups endorses patients' empowerment. *Patient Education and Counseling*, 74(1):61–69.

Verhoef, P. C. (2003). Understanding the Effect of Customer Relationship Management Efforts on Customer Retention and Customer Share Development. *Journal of Marketing*, 67(4):30–45.

Weiss, G. M. (2015). Mining with rarity: a unifying framework. *SIGKDD Explorations*, 6(1):7–19.

Yang, J., Wei, X., Ackerman, M., and Adamic, L. (2010). Activity Lifespan: An Analysis of User Survival Patterns in Online Knowledge Sharing Communities. *Icwsm*, pages 186–193.

YellowRoad (2017). Beware of Data Leakage: Don't say it will never happen to you.