UTRECHT UNIVERSITY

MASTER THESIS

Machine learning for predicting leads in content marketing

Author:
Jesse DE RUIJTER
Student number:
3837009

Supervisor:
Dr. A.J. FEELDERS
Second examiner:
Prof. dr. A.P.J.M. SIEBES

A thesis submitted in fulfillment of the requirements for the degree of Computing Science Master

August 14, 2018

Abstract

Jesse DE RUIJTER

Machine learning for predicting leads in content marketing

In this thesis I discuss the project I have worked on for the last 7 months. Commissioned by Jaarbeurs I created a model for predicting the number of leads specific content would generate in an online content marketing setting. I will describe how I addressed this problem and what methodology I used (Chapter 1). I will give an extensive overview of the data model I created and how I used imputation, feature engineering and feature selection to get the most out of the data (Chapter 2). In chapter 3, I will elaborate on the theoretical background of linear regression, logistic regression and survival analysis.

In chapter 4 the experiment setup and results of the models just using content data are discussed. A classification model is constructed to predict if a user would download certain content. This model is extended with features which describe a match between the user and the content (chapter 5). Survival analysis is used to make predictions depending on time. The newsletter data is added using time-dependent covariates (chapter 6).

In chapter 7, the results are discussed and a conclusion is drawn.

Acknowledgements

First of all, I would like to thank my supervisor Ad Feelders. He always gave meaningful suggestions and feedback. I would also like to acknowledge Wessel Vonk for supervising me in the work process, Jaring Hiemstra for giving me the opportunity to do this project, and all the people at Ynformed for helping me out and giving me tips!

Contents

Chapter 1

Introduction

1.1 Project context

1.1.1 Digital content marketing

Traditional marketing techniques are becoming less effective, as people consider them to be interruptive. Companies changed their stance from selling to helping. By sharing free content that could help customers, they try to bind customers to their brand (Holliman and Rowley, 2014).

Different types of content can be used to achieve different means. Depending on the goal of the company and the way they want to profile themselves, they will choose to share content of a specific type. Posting on social media, for example, increases brand awareness and enables a more personal form of communication and may allow a company to change the user's perception of their brand.

Another method is to involve users in the development process. By sharing blog posts with updates about the state of a new product, companies advertise their product before it is even releases. Moreover, the users will provide the company with useful feedback. Forums are often created with the same purpose. They help building a community and give an indication of how many people are interested in the product.

Also, whitepapers and e-books are published to increase the credibility of the company. Especially companies within research and technology fields benefit from having scientific content linked to their brand. Their authority on search engines will increase which makes it more likely for people to land on their website. Another purpose of sharing whitepapers and e-books is generating leads. People who interact with the content are more likely to buy the product.

1.1.2 Lead-generation process Jaarbeurs

Jaarbeurs has multiple online platforms (Computable.nl, Marqit.nl, Channelweb.nl, etc.) targeting different fields, e.g. IT, marketing, consultancy. Various types of content are shared with users on these platforms. Besides daily news items, blogs, reviews and columns the platforms have a knowledge-base with whitepapers and other scientific content. Companies can approach Jaarbeurs to do business-to-business (B2B) content marketing; content associated with their brand is shared on the platform. Another purpose is to generate leads. This process is illustrated in figure 1.1.

The company in question provides content for the marketing campaign. In most cases, a marketing campaign advertises 3-4 pieces of content, most often in the form of whitepapers or e-books. Users have to be registered to download and view content. When a user downloads content, this person is considered a potential lead for the company that supplied the content. Companies are able to set certain criteria

that the potential leads have to meet. Only if those criteria are met, the user can be returned as a lead. A criterion could be an attribute of the user, e.g. company size, branch or IT budget.

The platforms spread content among users. Additionally, users can subscribe to newsletters. There are several newsletters with different topics, linking to content on a certain platform. The newsletters will be sent out to users via e-mail in order to reach as many people as possible. Jaarbeurs keeps track of the people who download the content and this set of users is returned to the company as leads, after filtering out the users who did not meet the criteria.

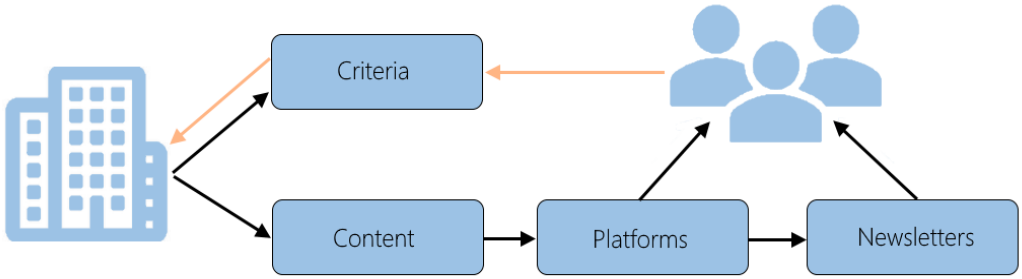


FIGURE 1.1: A schematic overview of the lead-generation process.

1.1.3 Project goal

Jaarbeurs decides on the price of B2B content marketing beforehand. They estimate the number of leads they think will be generated in the given time and they choose a price accordingly. Jaarbeurs guarantees a number of leads for a set price.

- If this guarantee is set too high, it won't be possible to generate enough leads in time.
- If the guarantee is set too low, it would have been possible to generate more leads in the given time and the prearranged price could have been higher.

For this reason, Jaarbeurs strives to predict the number of leads a content marketing campaign is going to generate, considering the content, criteria and newsletter planning. The main goal of this thesis is to create a prediction model tailored to the lead generation process of Jaarbeurs.

1.2 Research methodology

1.2.1 CRISP-DM

As the data is quite complex with its many attributes and underlying connections, it is hard to decide what model is the best fit. Therefore, a simple model will be used as starting point. Predictors are added to this model and the predictive value of this new model will be compared to the value of the base model. This way, the model with the best value will be selected to predict the number of leads a certain campaign is going to generate (Wirth and Hipp, 2000).

The CRISP-DM will be used as a guideline (figure 1.2). New insights will be gained at multiple points in the process: while reviewing data, evaluating the model

and at its deployment. By reiterating and implementing these new insights, the model will be improved during the whole the process.

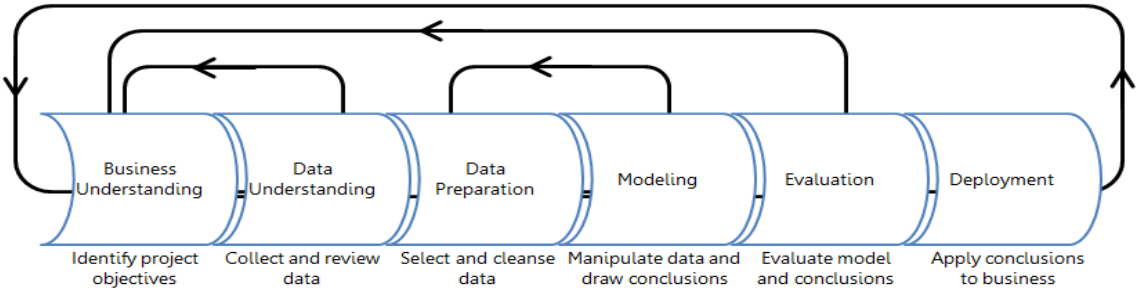


FIGURE 1.2: CRoss Industry Standard Process for Data Mining visualized as a pipeline.

1.2.2 Software

Programming language R

The R language is widely used among statisticians and data scientists for developing statistical software and performing data analysis. Data structures like vectors and matrices make it well equipped to efficiently deal with large amounts of data. The language has many built-in features for data preparation, creating models and statistically evaluating the outcomes of the models. As it is often used by researchers, many relevant packages are available with implementations of both well-known and lesser-known methods.

Packages

Some of the more important packages used for this project are:

dplyr, one of the more well-known packages. It adds data manipulation functions and a piping operator, which helps to make more concise, clean and readable code.

mice, a package for imputing data. Multiple methods for imputing are implemented and it gives you the option to choose per feature which method and data to use. It supports single and multiple imputation.

caret, or Classification And REgression Training. The caret package is a set of tools for building machine learning models in R. In this project it is specifically used for doing cross-validation and for its AIC stepwise logistic regression implementation.

glmnet, this package contains more extended generalized linear models. In this project it is used for its LASSO implementation.

survival, a package which implements survival analysis. It contains functions for preparing data for survival analysis and it also has multiple methods for performing it.

ggplot2, the most used package for doing visualization of data and statistics.

Chapter 2

Data overview and preparation

2.1 Data characteristics

2.1.1 Available datasets

There are six available datasets. Three of them contain entities: *users*, *content* and *newsletters*. The remaining three sets denote the link between these entities: *downloads*, *subscriptions* and *planning*. Each of these datasets and an overview of the features of each set can be found in the appendix.

Users

First of all, there is a file containing the complete userbase at 15-6-2017. This data set is a snapshot of the userbase at that time. There are no logs available of when a user was registered or removed, hence it is possible that other files point to a user which is not in the data anymore. Most likely, this will not occur often as users do not get removed frequently. A feature overview can be found in A.1.1.

Content

Besides data about users there is also data available about the content. The content is provided by the company that wants to use content marketing. Most content is in the form of whitepapers, but there are also reference cases and e-books. Content is published in one or multiple channels on one or more platforms. The channel is denoted in the data as *category*. It is possible that the same content gets posted on multiple platforms. A feature overview can be found in A.1.2.

Newsletters

The data about newsletters was already merged with the data of the subscriptions. This file contained a `user_ID`, a `newsletter_ID` and more information about the newsletters. I extracted the newsletters by anti-joining the unique newsletters. The left over subscription data was not useful, as it did not have the dates of the subscriptions included. I retrieved 128 newsletters from the data. It is possible though that there are newsletters I do not have any records of. A feature overview can be found in A.1.3.

Downloads

The download data contains information about which content is downloaded by which user at a certain date. It is also known how often a user downloaded the same

content. The oldest entry of this data is from 20-11-2014 and the latest is 5-7-2017. A feature overview can be found in A.1.4.

Subscriptions

I received another file with subscriptions. There is a column for every newsletter and a row for every user which registered or changed his subscriptions after 10-1-2013. The possible values for newsletter columns can be *0*, *1* and *NA*, where:

- the field has the value *0* if the user is not subscribed to the newsletter,
- the value *1* if the user is subscribed to the newsletter,
- or the value *NA* when the newsletter didn't exist at the moment that the user filled out his subscriptions, which means the user is not subscribed to the newsletter. It is also possible that the value is missing for a different reason, but I will assume the user is **not** subscribed to the newsletter.

There are 78 columns with a code which denotes the name of the newsletter. The names were manually linked to the names in the newsletter data. 76 out of the 78 newsletters were linked. The table was sparse as most people were only subscribed to several newsletters and many people were not subscribed to any of the newsletters. The data was transformed to have a row for every combination of user and newsletter the person is subscribed to. Also, this format is easier to work with. A feature overview can be found in A.1.5.

Planning

The data contains information about newsletters: when it was sent and what content was advertised in the newsletter. The data is divided over three files: one from 2015, one from 2016 and one from 2017. Every newsletter has its own excel-sheet. Not all files have the same newsletters, as not all newsletters were being sent in the same years.

The format of the data depends on the format of the newsletter. Every newsletter has a different layout and every newsletter in the data has a different structure depending on the actual layout. Because of this, there is information about the layout, but different parsers had to be written for many of the different newsletters.

- Information about the position of every paper within the newsletter was known for most of the newsletters, but not for all. In some newsletters the order of the data corresponds to the order of the papers in the newsletter. In other newsletters the order is denoted by a text field.
- If the newsletter had a text field denoting the order, then it also had information about how the content was displayed (only the title or the title and a description).
- The title of the edition of the newsletter was known for about half of the newsletters.
- Some papers in the newsletters are highlighted, they are put in a separate box with a different color, which should help the visibility of that specific paper. This is displayed by giving the row in the data a different color.

- Some of the newsletters have dynamic content. Depending on certain features of the users, they will receive different content. In this case there is a text field with the constraint of the group of users which will retrieve the newsletter.

At last, the data also contained ads which link to other websites. This is denoted by color-coding or the position field has the value "advertorial". A feature overview can be found in A.1.6.

2.1.2 Data structure

Combining all data, the following data model was constructed. This model provides an overview of the connection of the different data entities. It consists of three main entities: **newsletters**, **users** and **content**.

They are connected by the linking tables: **subscriptions**, **planning** and **downloads**. This creates a cyclic model, which could be used in multiple ways (figure 2.1).

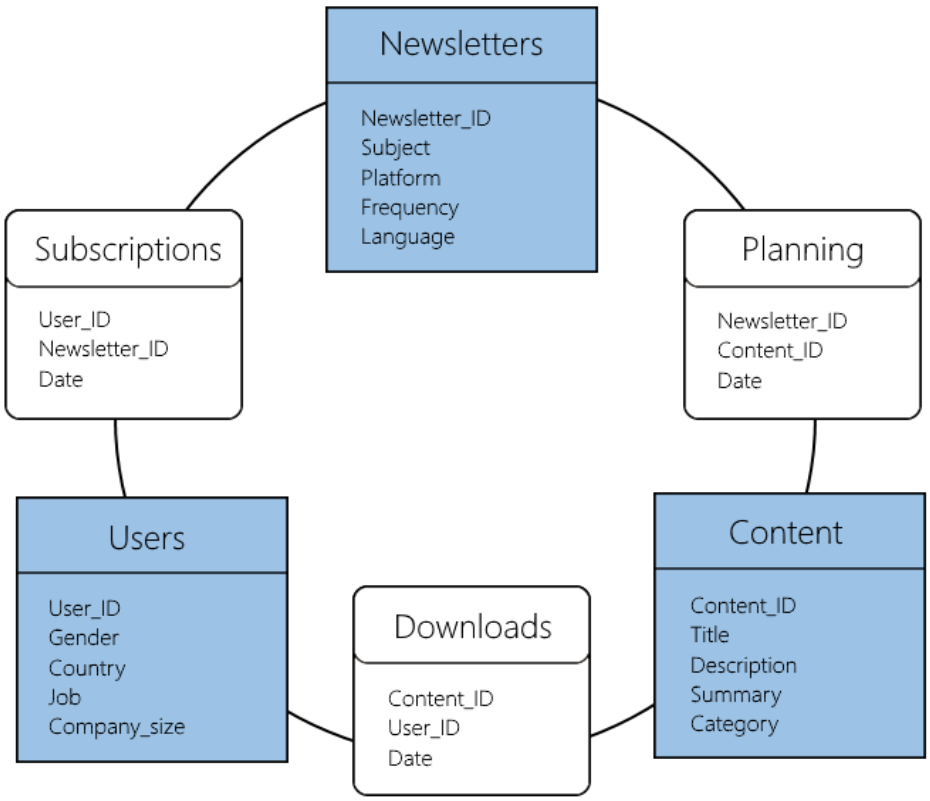


FIGURE 2.1: A visualization of the data model. The blue boxes are the entities and the white boxes are the linking tables.

In figure 2.2, the start and end date of the different datasets is plotted to give an overview of the time overlap. From 1-1-2015 to 15-6-2017 we have data from all sets, so this will be the time interval to be used.

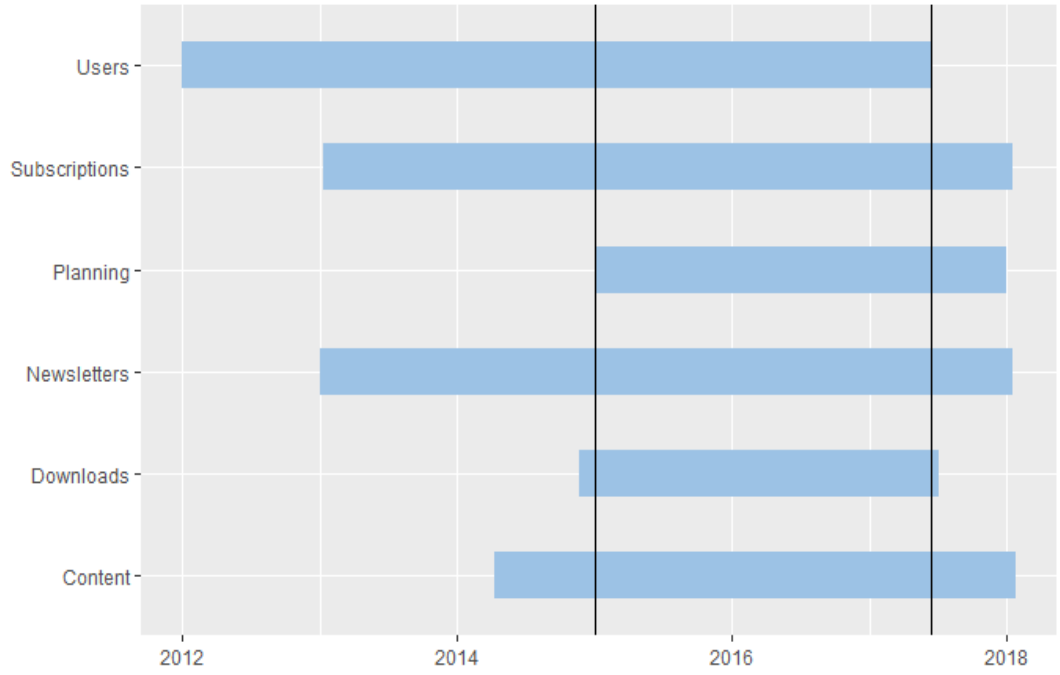


FIGURE 2.2: The time intervals of the available data sets.

2.2 Reducing the number of categories

Most machine learning methods will decode categorical variables with more than two levels as multiple binary categorical variables. If a variable has *n* levels, *n - 1* corresponding binary variables will be created with on average *1 / n* entries with value *1*. In conclusion, the higher the number of levels of a categorical variable, more binary variables with more skewed class distributions will be created under the hood, which is not beneficial for the model.

Some of the features of the data have many categories, so the number of levels was reduced using the following techniques:

- In some cases the levels were too specific and it was possible to combine multiple levels into an overarching one.
- If levels were not common enough and only occurred a couple of times, a level called *Other* was created and all levels with a support below a certain threshold was given that value.
- If there were only a few levels with a low support, they were combined with the most common level.

The following categorical variables had their levels reduced. The exact tables of the changed levels are included in the appendix.

Users

- **Gender**, there was a level *x* which denoted that the user did not want to share his or her gender. This level had a low support and these entries got value: *NA*.

- **Country**, all countries with a support lower than 30 were combined into the level *other*.
- **Branch**, multiple branches were combined into an overarching group (A.2.1).

Content

- **Category**, multiple branches were combined into an overarching group (A.2.2).
- **Platform**, some platforms only occurred a couple of times and they were placed in a group with remainders. Some platforms were separated on subcategories, they were combined into the main platform.

Downloads

- **Platform**, some platforms only occurred a couple of times and they were placed in a group with remainders. Some platforms were separated on subcategories, they were combined into the main platform.

2.3 Missing values

An important issue when preparing data is how to deal with missing values. Two of the available datasets had many missing values.

Users had missing values, caused by the registration procedure of the Jaarbeurs. The registration form changed overtime and there is a distinction between account information and profile information. Everyone who attends an event is required to complete the account registration. Only if someone wants to download content, it is also required to have the profile information completed.

Content also had many missing values, because their platforms were scraped for additional information and combined with the original content dataset.

2.3.1 Handling missing values

There are multiple ways of handling missing data. In some cases it is best to just remove the rows containing missing values. If a certain column has too many missing values it is possible to remove the entire column. The latter is not very beneficial in this case for the following two reasons:

- Columns were "fixed" using imputation.
- As feature selection was used: a column without useful information would not be used anyway.

Even though there is no scarcity of data, one should avoid throwing away data, because of the potential loss of useful data. Rows were only dropped in cases where they were not useful at all.

All users without an `user_ID` were left out, because there was no way of knowing what content they downloaded, as the download dataset uses this field as identifier. All content without a `content_ID` was left out for the same reason.

2.3.2 Imputation

All the other missing values were given a value by single imputation using the MICE-package (Van Buuren and Oudshoorn, 1999). Imputation is a method where a missing value is estimated by using the remaining data. Simple imputation methods will take for example the mean or mode of the feature, while more complex imputation methods use statistical methods or machine learning techniques to predict the value from the remaining data.

These different methods are all built-in the MICE-package. The following methods were applied to different types of features:

- *polr*, proportional odds logistic regression model for imputing ordered categorical values.
- *norm*, Bayesian linear regression for imputing numeric values.
- *polyreg*, multinomial logistic regression for imputing categorical values with more than two levels.
- *logreg*, logistic regression for imputing binary, categorical values.

At last, there is the option of using single imputation or multiple imputation and the number of used iterations must be decided. When using multiple imputation several datasets will be generated. These sets can be analyzed and combined into a final result. Although this method is proven to be more accurate, since it unnecessarily complicates the project, single imputation was used (Donders et al., 2006).

MICE uses an iterative algorithm. In general, the outcome converges after about 20-30 iterations. In this study, *maxit = 20* was used: this sets the function to 20 iterations.

2.4 Feature engineering

Feature engineering is creating more features out of existing data in order to get the most out of the data. Besides the obvious features that can be created by transforming data formats or by getting counts, additional information was also gained from the titles of the content and the match between some user and content features.

2.4.1 Simple features

- **Month**, the month content is published in can contain information about a reoccurring pattern. For example, many people tend to be on these platforms during working hours. In December most people have less working hours because of the holidays, so the average number of downloads in December is lower.
- **Weekday**, the weekday that content is published can also be of importance. When content is published it will be featured in the new items on the platforms. Being higher on this list makes the content more visible. Content posted when more people watch the platforms can be beneficial for the download rates.
- **Company count**, the total number of papers published by a company up to that date.

2.4.2 Title features

The content contains multiple text fields. The tags and summary can give a more precise description of the theme or subject of the content. The title and description of a paper contain more information. Besides the subject of a paper, people will also base their decision of reading certain content on how interesting the title is formulated. The same document could have different click rates given two titles. What features can describe the attractiveness of a title and can this information be used to make a better prediction of the click rate of a document?

- **Title length**, the number of characters in the title (Whissell, 1999).
- **Title list structure**, is the title formulated as a list. For example: *the top 10 things to do...*, *7 tips to make you better at...* or *You have been doing these 5 things wrong all along...*
- **Title question**, is the title formulated as a question? (Ball, 2009)
- **Title colon**, does the title contain a colon structure? (Whissell, 1999)

2.4.3 Matching features

We have a set of content C and a set of users U. For each content-user combination we make a pair with class y which has the value *1*, if the user downloaded the content and *0*, when the user did **not** download the content. When we want to predict how often a new piece of content which wasn't in C will be downloaded, we create a pair of this new content and every user in U and predict per pair if the user would download the new content or not. The sum of all these separate predictions will be our resulting prediction.

It's likely that certain features of the users, like how active the user was last quarter or education, will influence the result. Also there will be features of the content, like category or attractiveness of the title, which will influence the prediction. Both these features don't contain any information about the actual match between the users preferences and the content. It only tells us the likelihood of a user to download any content and the likelihood for content to be attractive to any user. Can we describe features which capture the match between two entities, which improve the accuracy of predictions made with the model?

Using feature engineering, I want to define features which can explain the match between a specific user and its interest in specific content (Joel, Eastwick, and Finkel, 2017). An interesting thing about the data is that this same pattern occurs three times.

- **Content – Downloads – Users**
- **Newsletters – Planning – Content:** Newsletters all have a topic. Is there a match between content and newsletters. Independently of the popularity of the content and the newsletter, does specific content do better in a specific newsletter?
- **Users – Subscriptions – Newsletters:** If the match between the user's interests and the topics of a newsletter fit well, does the user interact more with content?

The main focus will be trying to quantify to what extent the user's downloading behavior depends on the match between the user and the content. In order to do this

I will first construct a model containing just the content and the users. This model can be used as the control group. The next step is modeling the match and compare it with the base model to see if the match between user and content can improve the model.

- **Country user – language content**, is the language of the content one of the main languages in the country the user is from? It is highly unlikely that a person who doesn't understand dutch will download content written in dutch for example.
- **Work sector – category content**, does the sector a user works in match the subject of the paper? When a person works in healthcare is the person more likely to download content about healthcare?
- **Job category user – category content**, does the job category of the user match the subject of the content? This feature is similar to the previous one, but *job category* is more about the work you do (e.g. administrative assistant) while *work sector* is about the sector you are working in (e.g. logistics).
- **Category download history - category content**, has the user downloaded content with the same category as the new content in the past.
- **Language download history - language content**, has the user downloaded content with the same language as the new content in the past.

A reason why it is hard to make predictions is that most of the users never get to see most of the content. They might not be active on the platform the content was posted on or they are not subscribed to the newsletters where the content was mentioned in. If a user did not download certain content it does not necessarily mean that he wasn't interested in the content; it is also possible that the user never got to see the content.

- **Days till last logged in platform – platform published content**. Has the user been online on a certain platform since the content is published on the platform?
- **Newsletter**. This feature will describes if a user is subscribed to a newsletter which featured the content.

2.5 Feature selection

As mentioned before, the data contains many categorical variables with multiple levels. Therefore, the models have many coefficients, potentially overcomplicating the model. By utilizing feature selection, the following problems will be tackled:

- Making the model easier to interpret, by removing variables that are redundant.
- Reduce the number of coefficients to make the model run faster.
- Reduce overfitting.

The following method will be used for implementing feature selection.

2.5.1 LASSO

A well-known method for feature selection is *LASSO*, short for *Least Absolute Shrinkage and Selection Operator* (Zhao and Yu, 2006). This technique performs two main tasks to simplify a model: feature selection and regularization. LASSO sets a threshold on the sum of the model parameters. The method uses a shrinking process to shrink some of the parameters to zero penalizing the coefficients of the regression variables. This is tested with different parameter settings to minimize the prediction error. The features which didn't shrunk to zero are selected (Fonti and Belitser, 2017).

2.5.2 Implementation

At first, feature selection will be implemented only using LASSO. Multiple folds of training and test data were created for performing LASSO with cross-validation. *cv.glm* from the *glmnet* package was chosen: a function that aims to find an optimal λ by training multiple models using cross-validation. Thereafter it will set the coefficients of the variables that are not relevant to zero. The disadvantage of only using *LASSO* is that it will only keep one variable, `match_taal_history`. The importance of `match_taal_history` is higher than the importance of all other variables. Furthermore, the *LASSO* is not optimizing on our main objective, the number of downloads of content, but it will try to minimize the error of a classification problem.

This can be solved by tweaking the α-parameter which denotes for the weight between how much *Ridge* is used and how much *LASSO*. This method is called elastic net.

- For $\alpha = 0$, only Ridge is used.
- For $\alpha = 1$, only LASSO is used.
- For $\alpha > 0$ & $\alpha < 1$, a combination of both Ridge and LASSO is used called elastic net.

2.5.3 Results

An $\alpha = 0.05$ was used and this gave the following results.

In table 2.1 and table 2.2 there is an overview of all the features which got selected by the previously described method.

TABLE 2.1: The output of the LASSO model for the content.

Content		**Content**	
variable	**Estimate**	**variable**	**Estimate**
taal	-	category_ST	-0.0056
maand	-	category_ON	0.1714
type	0.0037	category_HI	-
weekdag	-	category_MA	-0.0273
titel_lijst	0.0105	category_OS	-0.0165
titel_vraag	0.0078	category_VI	-0.0639
titel_length	0.0022	category_IB	0.0406
titel_dubbelepunt	-	category_TL	-0.1069
category_IT	-0.0797	category_NC	-0.0054
category_FI	-	category_DI	0.0456
category_ZO	-0.0126	category_BO	-
category_BA	-0.0072	platform_ZONL	-
category_CC	-	platform_MQDK	-
category_ID	-0.0511	platform_V2DK	0.2983

TABLE 2.2: The output of the LASSO model for the users and matching features.

Users		**Match**	
variable	**Estimate**	**variable**	**Estimate**
land	-0.0042	match_datum	0.0002
geslacht	-	match_categorie_history	1.1330
opleiding	-	match_platform_history	1.1478
carriere_niveau	-	match_taal_history	6.0600
beroepsgroep	-0.0024	match_taal_land	0.0400
ict_budget	-	match_beroepsfroep_categorie	0.1069
gebruiker_type	-0.0012		
bedrijfsgrootte	-		
ictafdelingsgrootte	-0.0797		
leiding_aantal	-		
werkplekken_aantal	-		
betrokkenheid_inkoop	-		
branche	-		

Chapter 3

Theoretical background

3.1 Performance metrics

Several measures will be used to evaluate the performance of the models. Some measures are easy to interpret, while others reflect better how the model actually performed.

3.1.1 Mean absolute error

The mean absolute error, or MAE, measures the difference between the forecast values $\hat{Y}$ and the actual values Y. The absolute difference $|\hat{y}_t - y_t|$ is used to prevent them from canceling each other out (Willmott and Matsuura, 2005).

$$MAE = \frac{\sum_{t=1}^{T} |\hat{y}_t - y_t|}{T} \tag{3.1}$$

MAE is especially useful as a measure when compared to the MAE of the baseline model, a model which always predicts the average number of downloads of the training set.

3.1.2 Symmetric mean absolute percentage error

One of the main problems with MAE is that it only takes the absolute difference into account. For example, if the model would give a predicted value $\hat{y} = 301$ when the actual value is $y = 300$, then this would be a more impressive prediction than when $y = 20$ and $\hat{y} = 21$. Still both cases have the same MAE. This is why we introduce the mean absolute percentage error (MAPE) (Shcherbakov et al., 2013).

$$MAPE = \frac{100\%}{T} \sum_{t=1}^{T} \frac{|\hat{y}_t - y_t|}{y_t} \tag{3.2}$$

The problem with this metric is that overestimation results in a bigger penalty than underestimation, especially in our case. Most content will get around 20 downloads, but there are some really popular papers that get over 300 downloads. Falsely identifying a paper as really popular would give a MAPE of:

$$MAPE = \frac{100\%}{1} \sum_{t=1}^{1} \frac{|300 - 20|}{20} = 1400\%$$

while in the opposite case where you did not manage to catch a popular paper, the MAPE would be:

$$MAPE = \frac{100\%}{1} \sum_{t=1}^{1} \frac{|20 - 300|}{300} = 93.33\%$$

Because of this bias towards underestimating, an adjusted version will be used: symmetric mean absolute percentage error or SMAPE. It will use both the predicted value and the actual value for scaling the absolute error.

$$\begin{aligned} SMAPE &= \frac{100\%}{T} \sum_{t=1}^{T} \frac{|\hat{y}_t - y_t|}{|y_t| + |\hat{y}_t|} \\ &= \frac{100\%}{1} \sum_{t=1}^{1} \frac{|300 - 20|}{|20| + |300|} = 87.50\% \\ &= \frac{100\%}{1} \sum_{t=1}^{1} \frac{|20 - 300|}{|20| + |300|} = 87.50\% \end{aligned} \tag{3.3}$$

In formula 3.3 the same examples are calculated using SMAPE. Both examples have the same outcome. Also overestimation is not punished as much as with MAPE.

3.1.3 Normalized root mean squared error

If the number of downloads follows a Gaussian distribution with mean $\mu = 200$, the number of papers with 100 downloads is equal to the number of papers with 300 downloads. It would be desirable that predicting 99 instead of 100 gives the same error as predicting 301 instead of 300.

As can be seen in figure 3.1 when guessing lower numbers it is more likely to be right.

One last measure will be introduced: normalized root mean squared error or NRMSE for short. This is a variation on RMSE which takes either the range of the actual values into account ($max(y) - min(y)$) or the average of the actual values ($\bar{y}$). Division by the mean was chosen.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T} (\hat{y}_t - y_t)^2}{T}} \tag{3.4}$$

$$NRMSE = \frac{RMSE}{\bar{y}} \tag{3.5}$$

(Shcherbakov et al., 2013)

3.1.4 The area under a receiver operating characteristic curve

The last measure that will be introduced is *Area under the ROC curve* or *AUC*. Although we aggregate over the content, both the content-user models and the newsletter models are classification problems in essence. Besides using metrics to determine the error on the aggregated results, the classification error will also be considered.

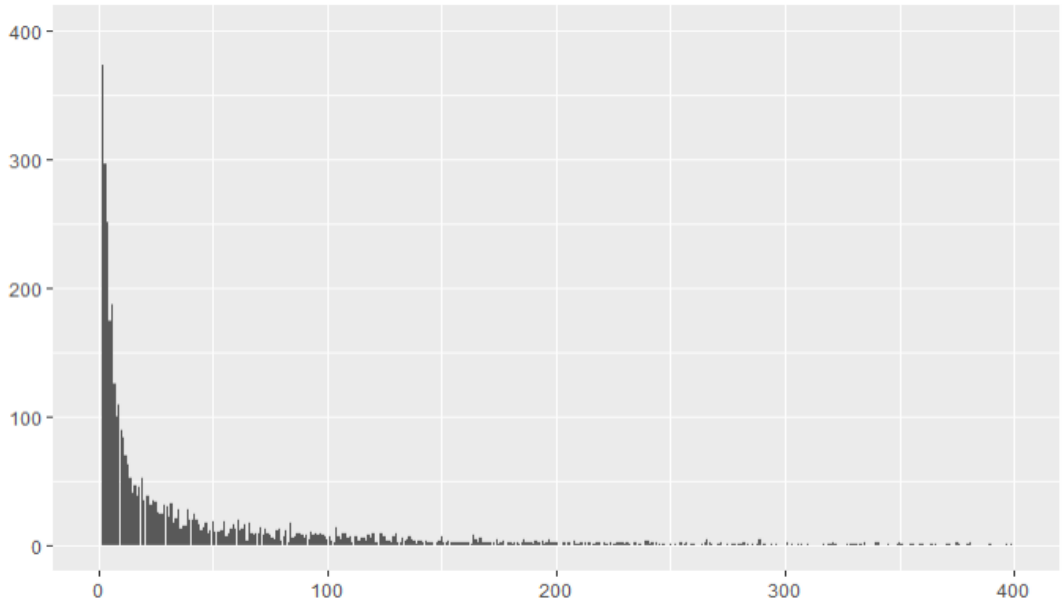


FIGURE 3.1: A plot with on the x-axis the number of downloads of the content and on the y-axis the count of how many papers have this number of downloads.

A receiver operating characteristic curve or ROC-curve is constructed by plotting $P(TP)$ versus $P(FP)$, where $P(TP)$ is the true positive rate and $P(FP)$ the false positive rate (figure 3.2). The predictions are sorted on their resulting probability $\hat{P}(y = 1)$ and the curve will rise when corresponding actual value belongs to class *1*. The AUC denotes how well the model separates the test set into the two classes. An AUC of *1* means that all entries in the test set which belong to class *1* were given a higher probability than the entries which have class *0*. While an AUC of *0.5* means that the model does not classify better than picking at random (Bradley, 1997).

3.2 Linear regression

For a subject i the goal is to get a prediction y_i, given a set of potential predictors of several types, $\{x_{i1}, ..., x_{ik}\}$. From these predictors we create a set of terms, the X-variables.

The general multiple linear regression model with response y_i and terms $\{x_{i1}, ..., x_{ik}\}$ has the form: (Weisberg, 2005)

$$\begin{aligned} E(y_i|\mathbf{X_i}) &= \beta_0 1 + \beta_1 x_{i1} + ... + \beta_k x_{ik} + \epsilon_i \\ &= \mathbf{B}^T \mathbf{X_i} + \epsilon_i \end{aligned} \tag{3.6}$$

In this formula $\mathbf{X_i}$ denotes the vector of X-variables and $\mathbf{B}^T$ is the transposed vector of parameters which has to be estimated. ϵ_i is the error term and is used to capture the remaining factors which influence y_i, which weren't already captured in the vector $\mathbf{X}_i$. The estimation of the parameters $\boldsymbol{B}$ can be done with multiple estimation techniques. The most common used ones are Ordinary least squares (OLS) and Maximum likelihood estimation (MLE).

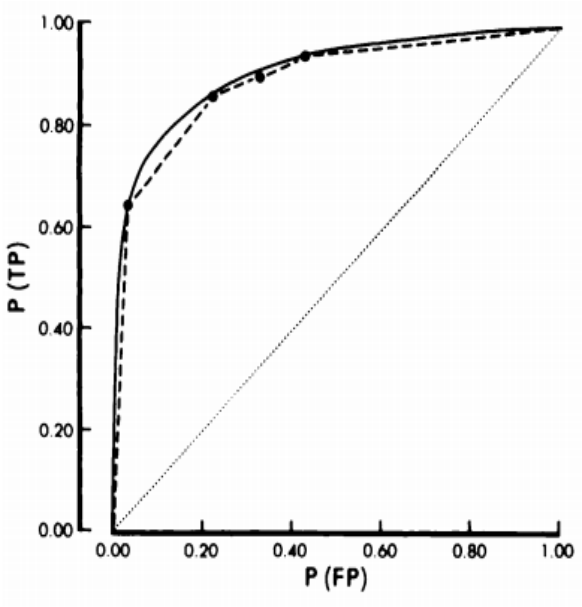


FIGURE 3.2: An example of a ROC-curve (Hanley and McNeil, 1982).

3.2.1 Ordinary least Squares

OLS is the most common method for estimating the parameters and it is also the method used by R's lm-function. Given formula 3.6, ϵ_i are independent random variables such that $E\epsilon_i = 0$. It is assumed that the relationship between dependent and explanatory variables is linear in parameters.

OLS minimizes the squared error and finds appropriate OLS estimators of β:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y \tag{3.7}$$

3.3 Logistic regression

In the case that the dependent variable is binary, logistic regression is a method which can be used. Linear regression is not suitable for predicting probabilities. For example, implementing a binary response variable in a linear regression model would not force it to give values between the *0* and *1* range. The output of linear regression can be transformed by using the logistic function, formula 3.8, which makes it suitable for probabilities.

$$p(x) = \frac{1}{1 + e^{-x}} \tag{3.8}$$

where x can be seen as a linear function $x = \mathbf{B}^T\mathbf{X_i}$. We can define $p(x)$ as the probability that the dependent variables are a success, $p(x) = p(y = 1|x)$.

$$logitp = \log\frac{p}{1-p} = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k \tag{3.9}$$

$$p(x) = \frac{1}{1 + e^{-\mathbf{B}^T\mathbf{X_i}}}$$
$$\frac{p(x)}{1 - p(x)} = e^{-\mathbf{B}^T\mathbf{X_i}} \tag{3.10}$$

(Menard, 2002)

3.3.1 Maximum likelihood estimation

The estimation method used for logistic regression is maximum likelihood estimation or MLE for short. Given the likelihood function $L(\theta|x)$

$$L(\theta|x) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i} \tag{3.11}$$

Or the log-likelihood which turns the product into a sum :

$$\begin{aligned} l(\theta|x) &= \ln L(\theta|x) \\ &= \sum_{i=1}^{n} y_i \ln p(x_i)(1 - y_i) \ln(1 - p(x_i)) \end{aligned} \tag{3.12}$$

In order to find the maximum likelihood estimates the log-likelihood function is differentiated with respect to the parameters and the derivatives are set to zero, $\frac{\partial l}{\partial \beta_j} = 0$ for each parameter β_j in $\boldsymbol{B}$. As it is usually not possible to solve this equation, the value is approximated.

3.4 Survival analysis

Another well-known method is survival analysis. Its main use is modeling the survival time of people or defect time for machines. A traditional application of survival analysis would be the following: a subject i, in this case a person, has the chance of getting heart disease. This is referred to as the event. A subject participates in the study for a certain period of time, the beginning of the observation period till the end. The end point of observation can be caused by multiple reasons:

1. the subject stopped participating,
2. the subject actually got the heart disease,
3. the study came to an end.

In the last case when a study came to an end and the event did not occur to a subject until that point the time is censored.

3.4.1 Proportional hazards model

Suppose that $\lambda(t)$ is the hazard function at time t. This function gives the chance for an event to occur for a subject i. The values $\mathbf{X_i}$ of individual i are denoted by $\{x_{i1}, ..., x_{ip}\}$ for p parameters. We try to find a set of parameters β, independent of

time, which describe the hazard of an event to happen at time t given the values $\mathbf{X_i}$ (Cox, 1992).

$$\begin{aligned}\lambda(t|\mathbf{X_i}) &= \lambda_0(t) \cdot e^{\left(\beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip}\right)} \\ &= \lambda_0(t) \cdot e^{\mathbf{B}^T \mathbf{X_i}}\end{aligned} \tag{3.13}$$

We are not interested in knowing the chance at a certain time, instead the main purpose of the model is knowing the chance of an event to occur up to time t given the predictors $\mathbf{X_i}$. For this reason we define the cumulative hazard function $\Lambda(t|\mathbf{X_i})$.

$$\begin{aligned}\Lambda(t|\mathbf{X_i}) &= \int_0^t \lambda(u|\mathbf{X_i})du \\ &= \int_0^t \lambda_0(u) \cdot e^{\mathbf{B}^T \mathbf{X_i}} du \\ &= e^{\mathbf{B}^T \mathbf{X_i}} \int_0^t \lambda_0(u) du \\ &= e^{\mathbf{B}^T \mathbf{X_i}} \Lambda_0(t)\end{aligned} \tag{3.14}$$

where $\Lambda_0(t)$ is the cumulative base hazard function equal to $\int_0^t \lambda_0(u)du$. The survival function $S(t|\mathbf{X_i})$ can be described using the cumulative hazard function.

$$\begin{aligned}S(t|\mathbf{X_i}) &= \exp\left[- \Lambda(t|\mathbf{X_i})\right] \\ &= \exp\left[- e^{\mathbf{B}^T \mathbf{X_i}} \Lambda_0(t)\right]\end{aligned} \tag{3.15}$$

This function has the following properties. $S(0|\mathbf{X_i}) = 1$, the function always returns *1* for $t = 0$, the survival chance is *1* at the start of the experiment. $S(\infty|\mathbf{X_i}) = 0$, the survival chance should converge to *0*.

3.4.2 Time-dependent covariates

In the previous section it was assumed that the predictors $\mathbf{X_i}$ did not change for a subject i over the course of the study. This is not always the case though. The predictors can be time-dependent and the values could change over time. When this is the case the simplification in formula 3.10 can not be made as it assumes $e^{\mathbf{B}^T \mathbf{X_i}}$ to be constant.

A way of dealing with time-dependent covariates is by splitting the parameters p in two sets (Kleinbaum and Klein, 2010):

- *p1*, the conventional parameters which are independent of time.
- *p2*, the time-dependent parameters. These parameters change over time as some event occurred at a certain point.

The hazard function is defined as follows:

$$\lambda(t|\mathbf{X}(t)) = \lambda_0(t) e^{\boldsymbol{\beta}^{p1} \mathbf{X}^{\mathbf{p1}} + \delta^{p2} \mathbf{X}^{\mathbf{p2}}(t)} \tag{3.16}$$

where $\boldsymbol{\beta^{p1}\mathbf{X^{p1}}} = \sum_{i=1}^{p1} \beta_i x_i$ and $\boldsymbol{\delta^{p2}\mathbf{X^{p2}}}(t) = \sum_{j=1}^{p2} \delta_j x_j(t)$. δ are the corresponding parameters similar to β, a different name is chosen to denote that it are time-dependent parameters.

The cumulative hazard function $\Lambda(t|\mathbf{X_i})$ is defined as:

$$\begin{aligned}\Lambda(t|\mathbf{X_i}) &= \int_0^t \lambda_0(u) \cdot e^{\boldsymbol{\beta^{p1}\mathbf{X^{p1}}} + \boldsymbol{\delta^{p2}\mathbf{X^{p2}}}(u)} \\ &\neq \Lambda_0(t) \cdot e^{\boldsymbol{\beta^{p1}\mathbf{X^{p1}}} + \boldsymbol{\delta^{p2}\mathbf{X^{p2}}}(t)}\end{aligned} \tag{3.17}$$

where $\Lambda_0(t) = \int_0^t \lambda_0(u)du$.

It is not possible to simplify the cumulative hazard function in the same way as the previous case, because the exponent has a side effect with a time-dependent value (Thomas and Reyes, 2014).

3.4.3 Time-dependent covariates

The main reason to use survival analysis is that it handles time very well. Another reason to use this method is to integrate the newsletters in the model. A user is subscribed to newsletters and content is advertised in these newsletters. Using a variation on time-dependent covariates we can model the event of a user receiving a newsletter with content.

There are two types of events that motivate a user to download certain content.

1. When content is published on a platform, it will be featured on the homepage of the platform it is posted on. This increases the visibility of the content and the chance of it being downloaded. When more new content is published it will move down on the page and subsequently the visibility will drop.

2. When the content did not generate enough leads it is featured in a newsletter to increase the visibility in the hope it will get more downloads. This can be seen as the other type of event.

3.4.4 Splitting the cumulative hazard function

The formula for time-dependent covariates as discussed in 3.4.2 might not be the best choice in this case. Especially when the predictors are not a function of time (the amount of days someone lived), but are interval-based (between t_1 and t_j the value is *3* and between t_j and t_n the value is *8*. Instead of introducing time-dependent parameters, it is also possible to split the hazard function at t_j.

Finding the cumulative hazard can be seen as finding the area under the base hazard function and multiplying it by the exponent of the predictors at that time. If the function is split at t_j, the area under the left and the right curve can be summed to get the area of the constructed function. In figure 3.3 an example of a base hazard function λ_0 is shown. In order to get the cumulative hazard $\Lambda(t_n|\mathbf{X_i}(\mathbf{T}))$ on t_n, the cumulative hazard up to t_j and the cumulative hazard from t_j to t_n can be summed. The predictor values $\mathbf{X_i}$ are constant within the intervals, thus the area of the left curve can be multiplied by $e^{\mathbf{B}^T\mathbf{X_i}(t_j)}$ and the area of the right curve has to be multiplied by $e^{\mathbf{B}^T\mathbf{X_i}(t_n)}$.

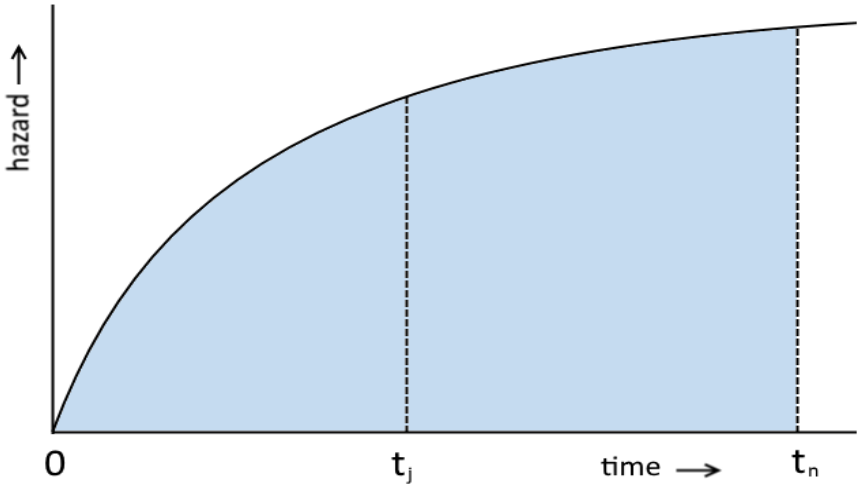


FIGURE 3.3: An example of a base hazard function, λ_0.

Say we want the survival function of subject i at time t. Subject i could have had multiple different events before t, thus there is a set of times $\{t_1, ..., t_n\}$, where n is the amount of events which is at least *1* and t_n is the latest time. The coefficients $\mathbf{X_i}(\mathbf{T})$ for each individual i differ for an event j: $\{\mathbf{X_i}(t_1), ..., \mathbf{X_i}(t_n)\}$ where each time t_j in $\mathbf{X_i}(t_j)$ is again a set $\{x_{i1}(t_j), ..., x_{in}(t_j)\}$.

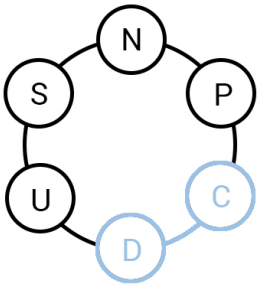
$$\begin{aligned}
\Lambda(t_n|\mathbf{X_i}(\mathbf{T})) &= \int_0^{t_1} \lambda(u|\mathbf{X_i}(t_1))du + ... + \int_{t_{n-1}}^{t_n} \lambda(u|\mathbf{X_i}(t_n))du \\
&= \int_0^{t_1} \lambda_0(u) \cdot e^{\mathbf{B}^T\mathbf{X_i}(t_1)}du + ... + \int_{t_{n-1}}^{t_n} \lambda_0(u) \cdot e^{\mathbf{B^T}\mathbf{X_i}(t_n)}du \\
&= e^{\mathbf{B}^T\mathbf{X_i}(t_1)} \cdot \int_0^{t_1} \lambda_0(u)du + ... + e^{\mathbf{B}^T\mathbf{X_i}(t_n)} \cdot \int_{t_{n-1}}^{t_n} \lambda_0(u)du \\
&= \sum_{j=1}^{n} \left(e^{\mathbf{B}^T\mathbf{X_i}(t_j)} \cdot \int_{t_{j-1}}^{t_j} \lambda_0(u)du \right)
\end{aligned} \tag{3.18}$$

$$\begin{aligned}
S(t_n|\mathbf{X_i}(\mathbf{T})) &= exp\Big[-\Lambda(t_n|\mathbf{X_i}(\mathbf{T}))\Big] \\
&= exp\Big[-\sum_{j=1}^{n} \left(e^{\mathbf{B}^T\mathbf{X_i}(t_j)} \cdot \int_{t_{j-1}}^{t_j} \lambda_0(u)du \right)\Big]
\end{aligned} \tag{3.19}$$

This formula holds, even if predictors are a function of time. When the intervals between the times in $\mathbf{T}$ have a limit that goes to zero, the value of the prediction approaches the actual value. In practice, the function is implemented as a step-function. The numeric predictor values are averaged over the time intervals. Especially in this case it is beneficial to implement it by splitting the cumulative hazard function.

Chapter 4

Content models



4.1 Content models

The first model uses just the content and download data. The download data is used to denote how often certain content is downloaded by counting the number of entries of a `content_ID` in the table. This is the most intuitive model. The number of downloads of new content has to be predicted, thus it is likely that the features of the content will be the most important predictors.

Multiple models using different techniques will be trained, tested and evaluated on several different datasets.

4.1.1 Experiment setup

As discussed earlier only download data from *1-1-2015* to *30-12-2017* is available, see figure 2.2. The download data is used to determine the number of downloads of the content. Therefore only the content which was published between *1-1-2015* and *30-9-2017* is used. Note that content which was published between *30-9-2017* and *30-12-2017* is not used to make sure the content has been online for at least 3 months.

Subset criteria

The company that supplied the content is able to give certain requirements that users should meet before they are considered a lead. If content is only used for predicting there is no possibility to know which downloads were from a user who met the requirements. The best option is to reduce the prediction with the same ratio as the requirements reduce the userbase.

The experiments are done on three different datasets:

1. *Users1*, the complete userbase.
2. *Users2*, a subset with only users who have a job related to IT.
3. *Users3*, a subset with only users who have an IT-budget above 500.000 euro.

For *users2* and *users3* the predicted number of downloads d is transformed using formula 4.1 to get the final prediction d'. U denotes all users in the training set and U_c are the users which meet the criteria c in the training set. D_u are all downloads for a user u.

$$d' = d \cdot \frac{\sum_{u \in U_c} D_u}{\sum_{u \in U} D_u} \tag{4.1}$$

In this formula $\frac{\sum_{u \in U_c} D_u}{\sum_{u \in U} D_u}$ compensates for the average number of downloads done by the users in the subset, $\sum_{u \in U_c} D_u$, compared to the number of downloads of whole userbase, $\sum_{u \in U} D_u$.

Representing platform and category

Content usually gets published on multiple platforms, thus the *platform* column in the content data is a list of platforms. It is also possible that the content is posted with a different *category* for each *platform*, so the *category* feature contains a list of categories for each row. These list formats are not convenient when creating a model. There are two possible ways of dealing with this problem. Both ways were tested.

1. One of the possibilities is to create a new row for each content-platform combination. When making a prediction, the `content_ID's` in the test set are used to find all content-platform combinations with that ID in the content. For each of these combinations a prediction is made and the predictions are summed over the ID's. The advantage of this method is that you know in what *category* the content was posted on each *platform*.

2. The other option is to make dummy variables for each *category* level and a dummy variable for each *platform* level. The content will have a binary variable for each *platform* and *category* whether it was posted there or not. The advantage of this approach is that the model has information about the combination of categories of the content.

Cross validation

10-Fold cross validation was used to validate the models. When using option 2 for handling the categories (as described in the previous section), the content got equally divided in 10 parts using *createFolds* of the caret package.

When using option 1 for handling categories, the caret package was used to divide the content into 10 parts of equal size. The `content_ID's` were used to group all content-platform combinations into training data or test data. The subsets do not have the same number of rows, because the contents of one subset can be posted on more platforms on average. All subsets do have the same count of contents. Three different models are trained using the resulting datasets: a linear regression model, a decision tree and a random forest. The predictions for each content-platform combination were grouped by content and summed. This way the predicted value could be compared to the actual number of downloads of the content.

4.1.2 Results

In table 4.1 the most important coefficients of the linear regression model on *users1* are displayed. The platform features seem to be the most important predictors with

high estimate values and very low p-values. Most of the categories were not relevant. This can be explained because many categories have high correlation with a specific platform. *Security* is one of the categories which is used on most of the platforms, which makes it more valuable.

TABLE 4.1: The most important coefficients of the content model.

variable	**value**	**estimate**	**std. error**	**t-value**	**Pr(>|t|)**
(Intercept)		117.30	20.70	5.67	1.48e-08
type	E-book	11.88	2.46	4.83	1.39e-06
type	Presentation	22.06	2.41	9.17	< 2e-16
type	Whitepaper	6.44	2.12	3.04	2.41e-03
language	English	-12.61	3.22	-3.92	9.08e-05
language	French	-15.94	5.60	-2.85	4.42e-03
language	Dutch	-9.00	3.29	-2.73	6.29e-03
date		-0.01	0.00	-5.67	1.50e-08
month	May	3.71	1.41	2.63	8.45e-03
month	November	4.22	1.63	2.60	9.46e-03
category	Cloud computing	2.18	0.78	2.78	5.39e-03
category	Education	8.91	1.81	4.92	8.83e-07
category	Security	6.64	0.78	8.50	< 2e-16
option 1					
platform	computable.nl	31.25	2.13	14.71	< 2e-16
platform	ict-en-logistiek.nl	15.39	2.64	5.83	5.67e-09
platform	infosecurity.nl	10.36	2.10	4.94	8.00e-07
platform	marqit.be	24.18	2.40	10.07	< 2e-16
platform	marqit.nl	46.19	2.12	21.80	< 2e-16
platform	not-online.nl	17.06	2.61	6.55	6.15e-11
platform	version2.dk	97.13	3.91	24.87	< 2e-16
platform	zorg-en-ict.nl	13.10	2.35	5.58	2.41e-08
option 2					
platform	channelweb.nu	21.51	5.19	4.15	3.44e-05
platform	computable.be	18.57	6.91	2.69	7.25e-03
platform	computable.nl	56.39	11.59	4.87	1.17e-06
platform	ict-en-logistiek.nl	19.96	4.58	4.36	1.33e-05
platform	infosecurity.be	13.36	3.20	4.18	2.95e-05
platform	infosecurity.nl	11.84	2.69	4.40	1.11e-05
platform	not-online.nl	27.99	4.18	6.70	2.35e-11
platform	overheid360.nl	10.22	3.80	2.69	7.19e-03
platform	version2.dk	134.19	10.55	12.73	< 2e-16
platform	zorg-en-ict.nl	21.62	3.21	6.73	1.95e-11

Tables 4.2, 4.3 and 4.4 contain the results of the experiments with the three different user datasets. Each table starts with the results of the baseline model which can be used for comparisson. In general over all three datasets the performance of the different models is similar, but random forest has the best performance followed by linear regression. The results also show that making separate entries for content on different platforms is a better method than creating dummy variables for all

categories (as described in *Representing platform and category*).

TABLE 4.2: The results of the content models with all users.

method	**dataset**	**platform**	**MAE**	**SMAPE**	**NRMSE**
baseline	users1	-	45.99	66.58	1.00
linear regression	users1	option 1	28.18	58.71	0.77
linear regression	users1	option 2	29.93	61.96	0.79
decision tree	users1	option 1	29.01	58.34	0.79
decision tree	users1	option 2	29.53	54.61	0.83
random forest	users1	option 1	29.12	54.40	0.78
random forest	users1	option 2	29.40	57.19	0.80

TABLE 4.3: The results of the content models with only users who have a IT-budget of 500.000 or higher.

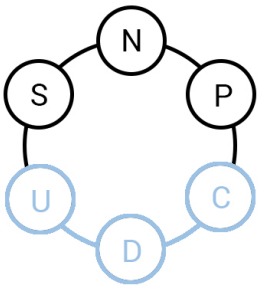
method	**dataset**	**platform**	**MAE**	**SMAPE**	**NRMSE**
baseline	users2	-	15.60	71.08	1.00
linear regression	users2	option 1	9.30	72.53	0.92
linear regression	users2	option 2	9.63	74.08	0.93
decision tree	users2	option 1	9.58	73.54	0.94
decision tree	users2	option 2	9.92	71.59	0.94
random forest	users2	option 1	9.12	69.92	0.91
random forest	users2	option 2	9.10	70.11	0.90

TABLE 4.4: The results of the content models with only users who have an IT-related job.

method	**dataset**	**platform**	**MAE**	**SMAPE**	**NRMSE**
baseline	users3	-	26.39	68.03	1.00
linear regression	users3	option 1	14.53	60.46	0.82
linear regression	users3	option 2	15.11	62.75	0.84
decision tree	users3	option 1	15.04	60.25	0.85
decision tree	users3	option 2	15.66	58.74	0.86
random forest	users3	option 1	14.03	58.28	0.79
random forest	users3	option 2	14.80	59.78	0.81

Chapter 5

Content-user models



In the previous chapter content models were discussed. In this chapter the content models are expanded by adding user data. Instead of using regression to predict the number of downloads, a classification model is built to predict if a user will download certain content. To estimate how often new content will be downloaded, a prediction is done for each user and the new content. The results are summed to retrieve an estimate of how often the new content will be downloaded.

The content variables are probably the better predictors and it is likely that the performance of the new models is similar to the content models. The main goal of adding users is to make the model more robust, as it is possible to pass different user sets to the model. For example, it is possible to use the user base at different moments (a year later new users will have registered). It also becomes possible to run the model with only users of a specific platform or only with active users (users who downloaded in the past 6 months or went to a Jaarbeurs convention).

Another upside to adding users to the model is that filtering users on the given criteria becomes an easier task. When doing regression with only the content, the output of the model is the number of downloads. There is no way of knowing which users downloaded the content. The best way of translating downloads to leads is looking at which part of the user base satisfies the criteria relative to the complete user base and use this same ratio. When using a content-user model the users who don not meet the requirements are filtered beforehand.

5.1 Logistic regression

5.1.1 Class imbalance

To get an idea of the class distribution, we will have a look at three datasets. There are 5313 entries in the content dataset, 385.263 users and 258.067 downloads. This makes $5313 * 385.263 = 2.046.902.319$ content user combinations. 258.067 of these combinations have the class *1* and $2.046.902.319 - 258.067 = 2.046.644.252$ have the class *0*. The ratio between the class labels is *1* to *7931*.

This causes two problems. Firstly, there are over two billion entries, which is not a feasible number to work with. The training data will have to be down-sampled to a better workable amount. Secondly, the class distribution *1* to *7931* is really skewed. If

we would sample in a way that the true class distribution is preserved, there would be too few cases where the user did download the content. It would be odd to train the model with a small number of downloads, especially because the main purpose of the model is detecting these cases.

Only down-sampling the cases with class label *0* will result in a better balanced class distribution in the training set. The data is sampled in a way that we are left with a class distribution ratio of *1* to *4*.

Case-control sampling

Although the model will do a better job capturing **which** people download **what** content, the probability estimates will be too high. The class distribution of the sample is not a good representation of the complete population. We can compensate for the sampling by adjusting the β_0 parameter as follows (Breslow and Cain, 1988):

$$\hat{\beta}_0^* = \hat{\beta}_0 + log\frac{\pi}{1-\pi} - log\frac{\tilde{\pi}}{1-\tilde{\pi}} \tag{5.1}$$

$\hat{\beta}_0^*$ is the adjusted intercept parameter.
π is class distribution of the whole population.
$\tilde{\pi}$ is the class distribution of the sample.

Compensating afterwards

Changing the intercept as described in the previous section is not a problem when using the standard *glm* function in R, because there is direct access to the coefficients. This makes it easy to change the intercept as described. Some problems arise when trying to do the same with the LASSO-variant *cv.glmnet*. It is not possible adjust the model coefficients, so the case-control sampling compensation has to be done after making a prediction.

When we plug in $\hat{\beta}_0^*$ from the previous section for $\hat{\beta}_0$ and solve:

$$\begin{aligned}
\frac{p(x)}{1-p(x)} &= e^{\hat{\beta}_0^* + \beta_1 x} \\
&= e^{\hat{\beta}_0 + log\frac{\pi}{1-\pi} - log\frac{\tilde{\pi}}{1-\tilde{\pi}} + \beta_1 x} \\
&= \frac{e^{log\frac{\pi}{1-\pi}}}{e^{log\frac{\tilde{\pi}}{1-\tilde{\pi}}}} e^{\hat{\beta}_0 + \beta_1 x} \\
&= \frac{\frac{\pi}{1-\pi}}{\frac{\tilde{\pi}}{1-\tilde{\pi}}} e^{\hat{\beta}_0 + \beta_1 x}
\end{aligned} \tag{5.2}$$

What we are left with is a factor depending on π and $\tilde{\pi}$. The predicted values have to be compensated by adjusting the odds with this factor c as follows:

$$\hat{p}(x) = \frac{cp(x)}{cp(x) + (1 - p(x))} \tag{5.3}$$

where $\hat{p}(x)$ is the adjusted probability.

Both methods were tested side by side and the results showed that the two methods do not give the exact same outcomes. The results are slightly lower when compensating afterwards, which in most cases also lowers the error.

5.1.2 Experiment setup

The outline of the implementation is as follows:

1. Create multiple training and test sets for cross-validation where the training data is down sampled as described in section 5.1.1.
2. Create the model using the training data and compensate for the class imbalance.
3. Use the model to make a prediction of the test data and aggregate the results to get the number of downloads of content.

In the next paragraphs it will be explained how the data sets are constructed and how the results are retrieved from the predictions.

Constructing the model

The method is shown in *Algorithm 1*. First of all the *caret* package is used to create k data folds on only the content which can be used for cross-validation. This creates two data sets $training_c$ and $test_c$. The users u and downloads d are joined. This resulting set will be joined with $training_c$ to create all the items in the training data with the class *1*, $training_1$.

In order to get the training data with class *0* the Cartesian product of the users and content needs to be sampled. The variable r is introduced to denote the ratio between the samples with class label *0* and *1*. In this case $r = 4$ is used. For each unique `content_ID` in the sampled content, as many users have to be sampled as the number of times that `content_ID` occurs in the sample. It is possible to get a content-user combination which was already in downloads. For this reason, all entries that are also in downloads are removed from the sample.

The removed entries should be re-sampled. The whole process could be placed in a loop with an exiting condition that r times the length of $training_1$ is reached. About 1 in 8000 entries should be repeated every time, so potentially many iterations are needed to get a result. This is why a max-depth could be used to ensure that the method will come to an end. I used a max-depth of only 1, because it is not important that the total length of the $training_0$ is exactly r times $training_1$. As long as the ratio $\frac{\pi}{\tilde{\pi}}$ is calculated from the sample and not using the variable r, there will be no problems.

For the test data just $test_c$ is used. The content of the test data is not joined with users as the resulting dataset is too large to fit into memory. For this reason the test data is only joined per paper just before making the prediction.

Predicting and results

After constructing the training and test data, a model is trained using the training data. A subset of the users is created with only the users who satisfy the criteria. For each entry in the test data, the content is merged with the subset just created. A prediction is made per content-user combination in the test set and the resulting probabilities are summed over the `content_ID`. This value is multiplied by the case control value to compensate for the class imbalance. The predictions for the number of downloads of content can be compared to the actual downloads and multiple performance metrics are calculated to describe the performance of the model.

Algorithm 1 Construct data sets

procedure CONSTRUCT TRAINING/TEST DATA
    $k \leftarrow 10$
    $r \leftarrow 4$

Set up folds for cross-validation:
    $(training_c, test_c) \leftarrow CreateFolds(k, c)$
    $ud \leftarrow Join(u, d, by=userID)$

    for i : 1 to k **do**
        $training_{true}[i] \leftarrow Join(trainingContent[i], ud, by = contentID)$.
        $lc \leftarrow r * length(training_1[i])$
        $sample_c \leftarrow Sample(sc, size = lc)$
        for all *j : Unique(contentID) in* $sample_c$ **do**
            $lu \leftarrow$*Count(j in* $sample_c$*)*
            $sample_u \leftarrow Sample(u, size = lu))$
        $training_0[i] \leftarrow Bind(sample_c, sample_u)$.
        $training[i] \leftarrow Bind(training_1[i], training_0[i])$.
        $training[i] \leftarrow RemoveDuplicates(training[i])$.

5.1.3 Results

Summed predictions

The final results are shown in table 5.1, 5.2 and 5.3. The baseline model is a model predicting the average number of downloads of the training data for all the content. The model which only uses the user features performs similar to the baseline model. This indicates that there is next to no predicting value in these features as expected.

The content and content-user models are both performing significantly better than the baseline model. The model using content and user features performs better on the third user set, while having similar performance on the other two user sets.

When comparing the models with feature selection versus the model without feature selection, no large differences in error are observed.

TABLE 5.1: The results of the classification models aggregated over the content only using the content features for users1.

model	dataset	feature selection	MAE	SMAPE	NRMSE
baseline	users1	-	47.85	65.48	1.00
content	users1	false	31.09	48.68	0.86
content	users1	true	30.97	49.09	0.83
user	users1	false	46.29	64.81	1.00
user	users1	true	46.44	64.86	1.00
content-user	users1	false	30.71	48.74	0.84
content-user	users1	true	30.99	48.58	0.84

TABLE 5.2: The results of the classification models aggregated over the content only using the content features for users2.

model	dataset	feature selection	MAE	SMAPE	NRMSE
baseline	users2	-	15.56	70.46	1.00
content	users2	false	9.05	59.28	0.79
content	users2	true	9.52	59.87	0.82
user	users2	false	14.90	69.20	1.00
user	users2	true	14.90	69.20	1.00
content-user	users2	false	9.19	59.25	0.83
content-user	users2	true	9.21	59.33	0.82

TABLE 5.3: The results of the content-user models aggregated over the content features for users3.

model	dataset	feature selection	MAE	SMAPE	NRMSE
baseline	users3	-	25.11	66.09	1.00
content	users3	false	16.59	53.55	0.98
content	users3	true	16.45	53.79	0.97
user	users3	false	23.53	65.45	1.00
user	users3	true	23.69	65.52	1.00
content-user	users3	false	15.57	53.44	0.89
content-user	users3	true	15.44	53.37	0.88

5.2 Matching features

The model is extended with features describing a match between the users and the content.

5.2.1 Experiment setup

A problem with implementing these features is that the product of content and users is too large to get into memory, as it has over two billion rows. For this reason the features are calculated for the training data and test data just before doing the predictions. This makes the process really slow especially on the available hardware.

There is a different option for some of the features. By making smart use of R's formulas the algorithm can be sped up. A formula has to be passed to the model, which denotes the variables that will be used for the model. Besides using variable names it is also possible to create combinations. For example, if there are two variables users $U \in \{1, 2\}$ and content $C \in \{3, 4\}$ using the formula $U : C$ will get all combinations between the two variables: $U1 + C3, U1 + C4, U2 + C3, U2 + C4$.

5.2.2 Results

The coefficients and p-values of the model are shown in table 5.4. The three features `match_platform_history`, `match_categorie_history` and `match_datum` have really low p-values, also in comparison to the other features. This gives the indication that the matching features might be relevant to the model as well.

TABLE 5.4: Matching features coefficients and p-values.

variable	value	estimate	Pr(>|t|)
`match_platform_history`	TRUE	3.2316	0.0000
`match_categorie_history`	TRUE	5.2700	0.0000
`match_datum`		0.0005	2.99e-13
`match_beroepsgroep_categorie`	TRUE	0.0302	0.6148
`match_land_taal`	TRUE	0.1222	0.6797
`match_taal_history`	TRUE	30.0044	0.8971

Classification

Before summing the predictions over the content, the quality of the classification per content-user combination is tested. This is done by creating a receiver operating characteristic curve or ROC-curve for short.

This method will arrange the predicted probabilities and check how well the corresponding actual classes are ordered. Because of this it is not a problem that the classes are imbalanced. The area under the curve or AUC can be used as measure for the quality of the predictions. If the model would be used for predicting the classes of the content-user combinations, then a threshold has to be determined. All entries with a predicted probability above this threshold will get the class *1*, entries with a value below the threshold get class *0*.

The resulting ROC's can be seen in figure 5.1. From left to right and top to bottom the models used are: *content*, *user*, *content-user* and *match*. The AUC's of these ROC-curves are given in table 5.5.

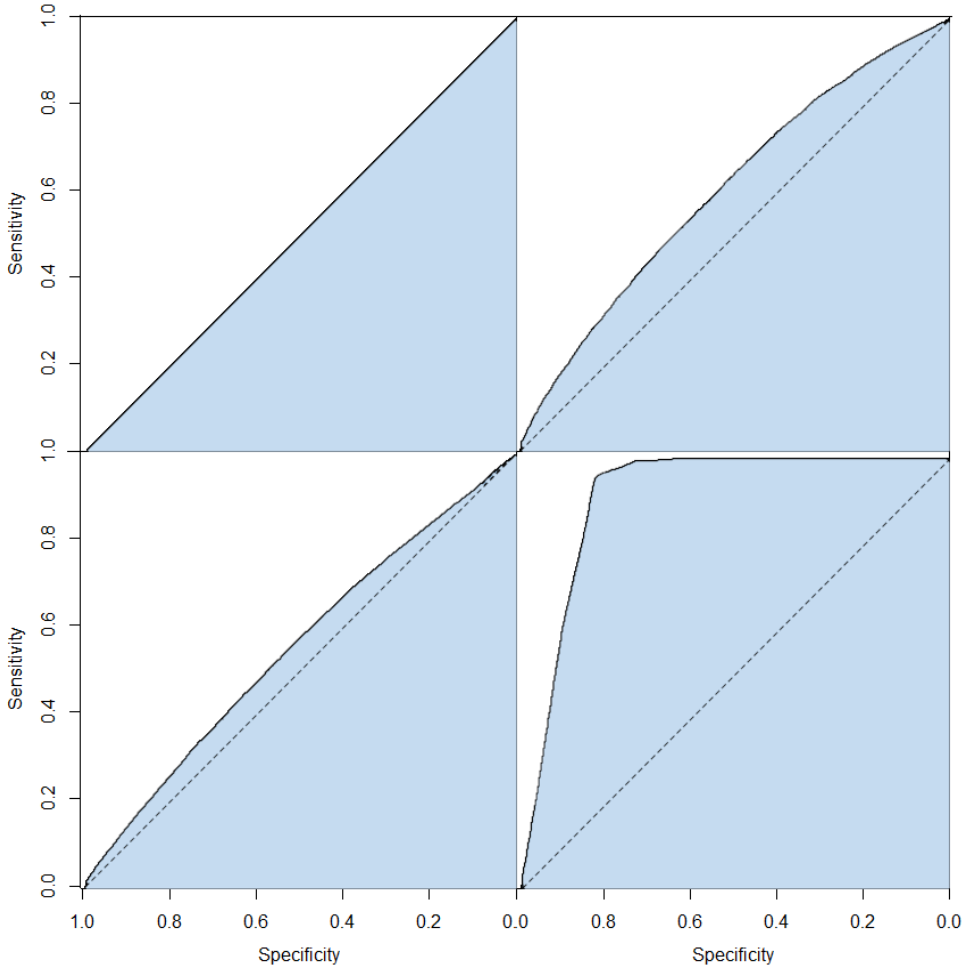


FIGURE 5.1: Area under the curve for the four different models.

TABLE 5.5: The results of the classification models aggregated over the content only using the content features.

Model	**AUC**
content	0.500
user	0.603
content-user	0.552
match	0.913

Using only the content to predict how well the model fits is not the optimal situation. Notably, the user model outperforms the content-user model. The model using the matching features shows the best results by far with an AUC of 0.919.

5.2.3 Summed predictions

Summing all the separate classification estimates gives the following results for the different datasets (tables 5.6, 5.7 and 5.8):

TABLE 5.6: The results of the content-user models aggregated over the content with content features, user features and the matching features.

model	dataset	feature selection	MAE	SMAPE	NRMSE
baseline	users1	-	47.85	65.48	1.00
logistic regression	users1	false	32.11	50.79	0.84
logistic regression	users1	true	32.62	50.15	0.86

TABLE 5.7: The results of the content-user models aggregated over the content with content features, user features and the matching features.

model	dataset	feature selection	MAE	SMAPE	NRMSE
baseline	users2	-	15.56	70.46	1.00
logistic regression	users2	false	9.33	60.67	0.81
logistic regression	users2	true	9.46	60.88	0.82

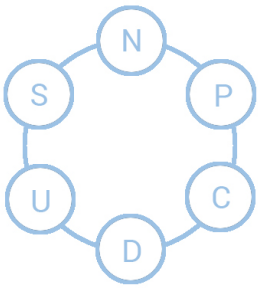
TABLE 5.8: The results of the content-user models aggregated over the content with content features, user features and the matching features.

model	dataset	feature selection	MAE	SMAPE	NRMSE
baseline	users3	-	25.11	66.09	1.00
logistic regression	users3	false	16.32	54.66	0.91
logistic regression	users3	true	15.94	54.61	0.89

The results are similar to the download estimates without matching features. The error is some what higher most likely due to overfitting. These results are unexpected as the AUC indicated that using matching features increases classification accuracy. This gain in accuracy didn't translate to improved estimation of the downloads. The AUC only measures how well the different classes can be divided, but the probability estimates might be a worse representation of the actual probabilities. This seems to be the case when using matching features.

Chapter 6

Newsletter models



6.1 Survival analysis

Survival analysis is a branch of statistics for analyzing the expected duration of time until one or more events happen. Its most common usage is to estimate the survival chance of a person, hence the name survival analysis. Although this paper has a completely different context, the model can be used as long as the data and the objective fit the model. Within this research the event can be seen as a user downloading content. The objective is to make a function for each content-user combination to determine the chance of a user downloading the content. The main advantage of using survival analysis is that this model is able to make predictions at different points in time. This allows us to estimate the downloads of certain content after two days, a week or a month.

The survival function will have the value 1 at time 0, because no one will have downloaded the content the moment it is published. Note that "surviving" is equal to "not downloading" in this case. After infinite time the survival rate should converge to 0. This is not of importance as the end of the study is at a set time. Another thing to note is that the function is stepwise, because it is an approximation of the actual survival rate. As the function is cumulative the curve can only decrease or remain the same at each step.

6.1.1 Experiment setup

The newsletters are implemented using time-dependent covariates as explained in section 3.4.4. A timeline of all events which happened to a user and the new content is constructed. The intervals between the events all get the same X-values. In order to create a timeline of the events of a user u and content c, users, subscriptions, newsletters, planning and content are joined to one large table. For each content-user combination only the entries with corresponding `content_ID` and `user_ID` are relevant. Also the following time restrictions have to apply:

$$t^{u}_{register} \leq (t^{c,u}_{subscribe})^{*} \leq (t^{c,u}_{newsletter})^{*} \leq (t^{c,u}_{download})^{+} \leq t_{censorRight}$$

$$t^{c}_{publish} \leq (t^{c,u}_{newsletter})^{*} \leq (t^{c,u}_{download})^{+} \leq t_{censorRight}$$

where $t^{u}_{register}$ is the registration time of the user. $(t^{c,u}_{subscribe})^*$ are the subscription times of the user to zero, one or more newsletters which advertise content c.$(t^{c,u}_{newsletter})^*$ are zero, one or more publishing times of a newsletter which mention the content c and user u is subscribed to. $(t^{c,u}_{download})^+$ denotes the possible event that the user downloads the content. Finally, $t^{c}_{publish}$ is the publishing time of the content. $t_{censorRight}$ is the censor time.

An example of a timeline is given in figure 6.1.

FIGURE 6.1: An example of an event timeline for a user u and content c.

The subscribe and register events are only used to check if the timeline is valid. The data structure is converted to time intervals where $X_i(\mathbf{T})$ are the values at the end of each interval.

6.1.2 Results

Four models were tested:

- a model using content and user features,
- a model using content, user and matching features,
- a model using content and user features with newsletters as time-dependent covariates as described in section 3.4.4,
- and at last a model using content, user and matching features in combination with newsletters.

The models were tested after different times: after 7 days, after 30 days and after 100 days. The AUC's of the models after 100 days can be found in figure 6.2 and the error of the summed predictions in table 6.1

The AUC's are similar to the AUC's of the logistic regression models while it is possible to make a prediction after a certain time with the survival analysis models. The AUC is higher if the model uses features describing a match between the content and the users. Using the time-dependent covariates does not significantly change the AUC.

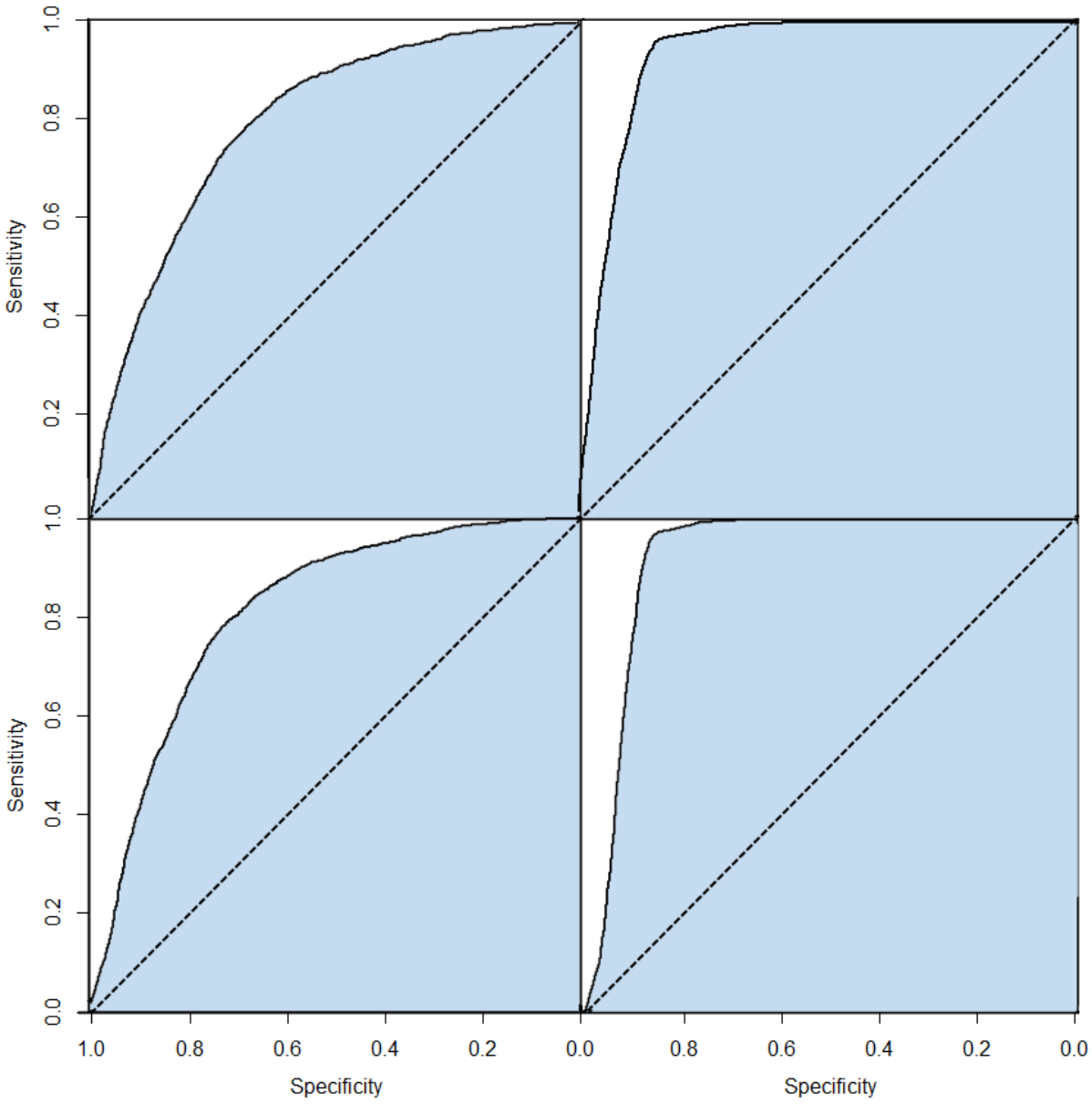


FIGURE 6.2: The resulting ROC-curves of the survival analysis on content-user models after 100 days.

TABLE 6.1: The results of the classification models aggregated over the content only using the content features.

Model	Newsletters	Days	MAE	SMAPE	AUC
baseline	-	7	0.992	86.65	0.5
content-user	false	7	0.703	85.53	0.749
match	false	7	0.623	82.11	0.902
content-user	true	7	0.846	84.14	0.765
match	true	7	0.893	83.90	0.893
baseline	-	30	2.253	73.27	0.5
content-user	false	30	1.358	72.41	0.760
match	false	30	1.090	70.16	0.924
content-user	true	30	1.488	71.42	0.772
match	true	30	1.284	68.96	0.913
baseline	-	100	3.160	69.81	0.5
content-user	false	100	1.638	66.19	0.783
match	false	100	1.197	62.68	0.938
content-user	true	100	1.917	65.55	0.792
match	true	100	1.548	62.15	0.927

Three conclusions can be drawn from the results shown in table 6.1:

- The AUC of the separate models does not differ much after different prediction times. The AUC gets slightly higher the longer the time interval used for the prediction. When making a prediction after 100 days, the AUC is the highest for all models, but not by much.

- The error of both performance metrics decreases if the time interval for prediction increases (the SMAPE decreases and the ratio between MAE and the MAE of the baseline as well). This is expected as the AUC increases. This can be explained by the fact that predicting after less days is harder as many users who might be interested, haven't had the opportunity yet to download it. The longer the content has been online, the more it is spread to the users by newsletters.

- A notable result is the SMAPE is lower for the models using newsletters (this indicates better performance), while the MAE is proportionally higher in comparison to the MAE of the baseline (this indicates worse performance). This result seems really strange, but can be explained by how the measures work. For the MAE it is beneficial to predict a value which is closer to the average number of downloads. Even if the case wasn't captured well by the model, the prediction has the highest chance to be close. The SMAPE divides the error by both the predicted value and the actual value (formula 3.3). Overestimating is less of a problem, because this will also increase the factor which the error is divided by.

  What we can take away from this is that using the time-dependent covariates the predictions will be less reserved, which causes it to have a higher absolute error.

Chapter 7

Conclusion

7.1 Conclusion

Commissioned by Jaarbeurs a prediction model was created which can estimate the number of leads new content will generate. This is valuable for Jaarbeurs, as it can help with the negotiations of B2B content marketing by giving an indication of how many leads can be generated in a certain time. The models were compared to a baseline model which always predicts the average number of downloads of the content in the training data. When interpreting the results we have to take into account that the content managers at Jaarbeurs have a better idea of how well certain content will perform from experience. Also the content managers can influence the number of downloads content will receive after being published by advertising the content in newsletters.

I started out by creating models which only use the content data (chapter 4). These models had no real way to deal with time, the prediction was done 100 days after publishing the content. Several experiments have been performed on the content data in order to validate the different models and methods. I looked at two different ways of structuring the data: the same content on different platforms as separate rows or one row for every piece of content and binary variables for the platforms. The results of the experiment showed that structuring the content data in separate rows is the best option. This shows that it is more useful to know in what category content was published on a specific platform than knowing the combination of categories or platforms (table 4.2, 4.3 and 4.4). Random forest had the best performance out of the tested methods with a Mean Absolute Error of *29.12* on *users1* in comparison to the baseline model with an error of *45.99*. This is a significant decrease in error, however it is hard to say how valuable this prediction is in practice for multiple reasons. As mentioned before, the employees at Jaarbeurs probably make better predictions than the baseline model. The model will most likely get less relevant as the userbase changes and the number of users which pass the criteria is estimated. At last, the model always does predictions after 100 days which makes it less useful in practice.

In chapter 5, I constructed a different model which predicts if a user would download new content. By summing over the predictions of all the users which meet the user requirements, an estimate of the total number of downloads is given. This method could potentially be more robust considering changes in the userbase and requirements. When only using user data the model performed similar to the baseline model, which indicates that all the relevant information is in the content features. Using a model with both content and user data seems to be slightly better at handling different user sets, especially user set 3 (table 5.3). This gain in performance does not outweigh the drop in performance caused by using classification.

Adding features which describe a match between user and content greatly increase the performance of the classification. The AUC is significantly higher (figure 5.1 and table 5.1) which indicates that the accuracy increases by adding these matching features. Despite the increase in AUC the actual prediction of the number of downloads isn't better than the previous predictions (table 5.6, 5.7 and 5.8). This is possible if the probability estimates are more suitable for separating the data in two classes, but are not a better presentation of the actual chance of being downloaded.

Survival analysis turned out to be a suitable method for handling the time component of the problem. Different models including content features, user features, features describing a match between the two, and newsletters features were used for experiments. The models had to make a prediction after 7, 30 or 100 days. The predictions after 100 days had the best AUC (figure 6.2) and the lowest error (table 6.1), which can be explained as follows. Users have had more time to download certain content after 100 days, which makes it more likely that people who are interested in the content have already seen it and downloaded it. Adding newsletters through time-dependent covariates is an interesting idea. I think this approach has potential for modeling events in cases similar to this one. In this specific case no major benefits have been achieved by adding the newsletters, which can be partially attributed to the poor data quality of the newsletters.

All things considered, a program was delivered which implements a model based on survival analysis techniques. This program still has to prove its worth in actually increasing profit by helping during price negotiations.

7.2 Future work

I showed in this thesis that with feature engineering a lot of improvements can be made to a classification model. If the structure of the data is similar to the Jaarbeurs data with two tables which both will be subset for making predictions, then using a similar approach might make the model more robust. The problem with the available data is that it is too one-sided. When classifying if a user will download certain content, the content features were more significant. This is because most information about users was not supplied because of privacy reasons. Features like age, the company the person was working at, city, etc. could have had a lot of value for the prediction. Also the newsletter data was manually kept up, which made its quality fairly low. If the infrastructure of Jaarbeurs would be upgraded in a way that this data is integrated in the system and saved in a database, then the value of the newsletters in the model could be a lot higher. This might happen in the near future.

Secondly, experimenting with more different models for classification could be interesting. When using different models, do the matching features also increase the performance or did the model already find the connections without using the matching features. Also the gain in performance measured by the AUC can be better translated to a lower error on the regression problem.

Appendix A

Tables and figures

A.1 Complete overview of the data and features

A.1.1 Users

TABLE A.1: All the features of the content listed with their attribute types, levels and a description.

Feature	Attribute type	Levels	Description
User_ID	Numeric	-	-
Country	Categorical	24	-
Gender	Categorical	2	-
Platform	Categorical	20	The website where the person registered.
Education	Ordinal	5	Highest level of education according the Dutch education system.
Professional_level	Categorical	6	The work experience of the user.
Professional_group	Categorical	17	The field of the users job.
Professional_subgroup	Categorical	49	A more precise indication of the field.
IT_budget	Ordinal	10	The available budget for IT of the users company.
User_type	Categorical	6	End user, adviser, reseller, supplier.
Department	Categorical		
Company_size	Ordinal	13	The number of people working in the user's company.
IT_department_size	Ordinal	8	The number of people working within the IT department.
Team_lead_size	Ordinal	6	The size of the team lead by the user.
Workplace_size	Ordinal	7	Number of workplaces available in the user's company.
Function	Text	-	-
Purchasing_involvement	Categorical	8	The purchasing rights of the user within his company.
Branch	Categorical	22	

A.1.2 Content

TABLE A.2: All the features of the content listed with their attribute types, levels and a description.

Feature	Attribute type	Levels	Description
`Content_ID`	Numeric	-	-
Title	Text	-	Title of the content.
Summary	Text	-	A summary of the content written by the Jaarbeurs employees.
Description	Text	-	An extensive description provided by the company.
Tags	Text	-	Tags written by the employees of Jaarbeurs.
Category	Categorical	17	The category of the content.
Type	Categorical	18	E-book, whitepaper, reference case, etc.
Company	Text	-	The company which supplied the content.
Date	Date	-	The publication date of the content.
Language	Categorical	4	The language the content is written in.
Platform	Categorical	20	The website where the content is published on.

A.1.3 Newsletters

TABLE A.3: All the features of the content listed with their attribute types, levels and a description.

Feature	Attribute type	Levels	Description
`Newsletter_ID`	Numeric	-	The ID of the newsletter.
Platform	Categorical		The website the newsletter is sent from.
Name	Text	-	The name of the newsletter.
Code	Text	-	A code denoting which newsletter it is (redundant).
Language	Categorical	3	The language the newsletter is written in.
Country	Categorical	3	The country the newsletter is meant for.
Subject	Categorical	24	The subject of the newsletter.
Frequency	Numeric	-	How often the newsletter is sent.

A.1.4 Downloads

TABLE A.4: All the features of the content listed with their attribute types, levels and a description.

Feature	Attribute type	Levels	Description
`User_ID`	Numeric	-	ID of the user downloading the content.
`Content_ID`	Numeric	-	ID of the content being downloaded.
Date	Date	-	The date when the content was downloaded for the last time by a user.
`Download_count`	Numeric	-	The number of times the same user downloaded the same content.

A.1.5 Subscriptions

TABLE A.5: All the features of the content listed with their attribute types, levels and a description.

Feature	Attribute type	Levels	Description
`User_ID`	Numeric	-	The ID of the user subscribing.
`Newsletter_ID`	Numeric	-	The ID of the newsletter being subscribed to.
Date	Date	-	The date when the user subscribed to the newsletter.

A.1.6 Planning

TABLE A.6: All the features of the content listed with their attribute types, levels and a description.

Feature	Attribute type	Levels	Description
`Content_ID`	Text	-	The ID of the content which is advertised.
`Newsletter_ID`	Text	-	The ID of the newsletter the content is in.
Position	Ordinal	28	The place of the content within the newsletter.
`Content_title`	Text	-	The title of the content.
Date	Date	-	The date when the newsletter was sent.
featured	Categorical	2	Boolean to denote if the content was highlighted in a special section.
`Dynamic_content`	Categorical	-	The group the content was sent to.
`Edition_title`	Text	-	The title of the edition of the newsletter.

A.2 Overview of reduced features

A.2.1 Branch

TABLE A.7: The previous values of branch and their new values.

New branch	**Previous branch**
Gezondheidszorg/Welzijn	Gezondheids- en welzijnszorg - Zelfstandig Behandel Centra (ZBC) Gezondheids- en welzijnszorg - Apotheken Gezondheids- en welzijnszorg - Jeugdzorg en Jeugdzorg instellingen Gezondheids- en welzijnszorg - Toeleverancier gezondhiedszorg Gezondheids- en welzijnszorg - Thuiszorg; welzijn en ouderenzorg Gezondheids- en welzijnszorg - Geestelijke gezondheidszorg Gezondheids- en welzijnszorg - Huisartsen Gezondheids- en welzijnszorg - Gehandicaptenzorg Gezondheids- en welzijnszorg - Private Klinieken Gezondheids- en welzijnszorg - Verpleeg- en verzorgingshuizen Gezondheids- en welzijnszorg - ziekenhuizen Gezondheids- en welzijnszorg - Fysiotherapie/revalidatie Zorg - 1e lijns praktijk Zorgverzekeraar Zorggroep
Onderwijs/Opleiding	Onderwijs - Wetenschappelijk Onderwijs (WO)) Onderwijs - Primair onderwijs Onderwijs - Overig Onderwijs Onderwijs - Particulier Onderwijs - Hoger Beroepsonderwijs (HBO) Onderwijs - Primair speciaal onderwijs Onderwijs - voortgezet onderwijs (havo/vwo) Onderwijs - Nieuwe Lerarenopleiding (NLO) Onderwijs - Pedagogische Academie Basis Onderwijs (PABO) Onderwijs - voorbereidend middelbaar beroepsonderwijs (vmbo) Onderwijs - voortgezet speciaal onderwijs Onderwijs - Kinderopvang/BSO Onderwijs - MBO/BVE
Overheid	Overheid/Non-profit Overheid - Gemeente - A&I Overheid - Gemeente -WMO Overheid - Gemeente - Ruimte en GEO Overheid - Regionale Uitvoeringsdiensten/Omgevingsdiensten Overheid - Gemeente - Samenlevingszaken Overheid - Politie / brandweer Overheid - Gemeente -Publiekszaken Overheid - Gemeente

TABLE A.8: The previous values of branch and their new values.

New branch	**Previous branch**
Overheid	Overheid - Provincie Overheid - Waterschap Overheid - Rijksoverheid
Industrie	Vervaardiging van metalen (producten) Vervaardiging van voedingsmiddelen en dranken Vervaardiging van producten van rubber en kunststof Metaalindustrie Vervaardiging van chemische producten (incl. farmaceutische industrie) Techniek
ICT - leveranciers hardware	Elektronica
Vrijetijdsbesteding	Media/Uitgeverijen/TV Horeca/Toerisme/Recreatie/Sport (Openbare) bibliotheken Kunst/Cultuur/Entertainment
Dienstverlening	Facilitaire Dienstverlening Banken/Financiële dienstverlening Juridische dienstverlening Overige dienstverlening Uitzend/Detachering/W&S
Zakelijke dienstverlening	verzekeringen/Assurantie Accountancy
Handel/Groothandel	FMCG (Fast Moving Consumer Goods) Online retail Mode/Textiel/Cosmetica Landbouw/Bosbouw/Visserij Makelaardij/Vastgoed
Transport/Opslag/Distributie	Automotive Maritiem
Bouw/Installatie	Energie/Gas/Water

A.2.2 Category

TABLE A.9: The previous values of category and their new values.

New category	**Previous category**
Bouw	Gevelsystemen Gevelopeningen Gevelbekleding Gevelverlichting Renovatie Isolatie Coatings Aanbouwdelen en componenten Dak Gevels Opslagsystemen Hijswerktuigen en kraansystemen
Business applicaties	Business Applicaties
Cloud computing	Cloud Computing
Diensten	Advies en organisatie Diensten en advise E-fulfilment services Webdiensten Brancheorganisaties Producten en diensten ZorgTotaal Logistieke dienstverlening
Facilitair & Inrichting	Gebouwgebonden inrichting Huisvesting & facilitair Facilitair en Inrichting Domotica Comfort Klimatisering
Hardware & Infrastructuur	Embedded systems & Robotics 3D printing Automatisering en Automatische Identificatie Microsystems Components & Technology Hardware en Infrastructuur Batterijen Energie management & electrische systemin Energie Datacenters Robotica Industrial Automation Systems & Handling Systems
Informatie & Data management	Informatie en data management Informatiebeleid & ICT IT Beheer Management Sales & Operations Planning

New category	**Previous category**
Informatiebeveiliging	-
Maatschappij	Burger & Bedrijf Leefomgeving & Veiligheid Mens & Milieu Theater en Cultuur Connect Control Overheid specifieke software
Netwerk & Communicatie Onderwijs	Netwerk en Communicatie Leren en Onderwijzen met ICT Opleiding en Training Zorg & Onderwijs Het Jonge Kind VO PO MBO Dagtrips
Outsourcing & Software ontwikkeling	ICT & Outsourcing in de zorg Industrial IT & Software Operating Systems
Transport & Logistiek	Transportbanden, -goten en -banen Hef- en magazijntrucks Verpakkings- en handlingsystemen Handbediend magazijn materiel Laad- en losapparatuur Containers, vaten, kratten, big bags, pallets Magazijn & Distributie Logistieke software Hulpmiddelen Identificatie (Auto-ID) Logistiek en veiligheid Logistiek Vastgoed Transport
Virtualisatie	Virtualisatie
Zorg	Samenwerken in de zorg Kwaliteits- en Kennismanagement in de zorg Samenwerken in de zorg Thema's ZorgTotaal Smart Health Care

Bibliography

Ball, Rafael (2009). "Scholarly communication in transition: The use of question marks in the titles of scientific articles in medicine, life sciences and physics 1966–2005". In: *Scientometrics* 79.3, pp. 667–679.

Bradley, Andrew P (1997). "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern recognition* 30.7, pp. 1145–1159.

Breslow, N. E. and K. C. Cain (1988). "Logistic Regression for Two-Stage Case-Control Data". In: *Biometrika* 75.1, pp. 11–20. ISSN: 00063444. URL: http://www.jstor.org/stable/2336429.

Cox, David R (1992). "Regression models and life-tables". In: *Breakthroughs in statistics*. Springer, pp. 189–190.

Donders, A Rogier T et al. (2006). "A gentle introduction to imputation of missing values". In: *Journal of clinical epidemiology* 59.10, pp. 1087–1091.

Fonti, Valeria and Eduard Belitser (2017). *Feature Selection using LASSO*.

Hanley, James A and Barbara J McNeil (1982). "The meaning and use of the area under a receiver operating characteristic (ROC) curve." In: *Radiology* 143.1, pp. 29–36.

Holliman, Geraint and Jennifer Rowley (2014). "Business to business digital content marketing: marketers' perceptions of best practice". In: *Journal of research in interactive marketing* 8.4, pp. 269–293.

Joel, Samantha, Paul W Eastwick, and Eli J Finkel (2017). "Is romantic desire predictable? Machine learning applied to initial romantic attraction". In: *Psychological science* 28.10, pp. 1478–1489.

Kleinbaum, David G and Mitchel Klein (2010). *Survival analysis*. Vol. 3. Springer, pp. 211–255.

Menard, Scott (2002). *Applied logistic regression analysis*. Vol. 106. Sage.

Shcherbakov, Maxim Vladimirovich et al. (2013). "A survey of forecast error measures". In: *World Applied Sciences Journal* 24, pp. 171–176.

Thomas, Laine and Eric M Reyes (2014). "Tutorial: survival estimation for Cox regression models with time-varying coefficients using SAS and R". In: *J Stat Softw* 61.c1, pp. 1–23.

Van Buuren, Stef and Karin Oudshoorn (1999). *Flexible mutlivariate imputation by MICE*. Leiden: TNO.

Weisberg, Sanford (2005). *Applied linear regression*. Vol. 528. John Wiley & Sons.

Whissell, Cynthia (1999). "Linguistic complexity of abstracts and titles in highly cited journals". In: *Perceptual and Motor Skills* 88.1, pp. 76–86.

Willmott, Cort J and Kenji Matsuura (2005). "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance". In: *Climate research* 30.1, pp. 79–82.

Wirth, Rüdiger and Jochen Hipp (2000). "CRISP-DM: Towards a standard process model for data mining". In: *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*. Citeseer, pp. 29–39.

Zhao, Peng and Bin Yu (2006). "On model selection consistency of Lasso". In: *Journal of Machine learning research* 7.Nov, pp. 2541–2563.