Automated classification of user reviews.

Detection of topic and sentiment

by

Jos Steijn

to obtain the degree of Master of Science in Applied Mathematics

at the Delft University of Technology,

to be defended publicly on Tuesday May 26th, 2020 at 15:00 PM.

Student number:	4281667	
Project duration:	September 2, 2019 – May 12, 2020	
Thesis committee:	Prof. dr. ir. G. Jongbloed,	TU Delft, supervisor
	Dr. ir. G. F. Nane,	TU Delft
	Drs. J. W. Bikker PDEng,	CQM

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Consultants in Quantitative Methods

Abstract

Online customer reviews on products have become a large part of marketing intelligence in recent years. These documents are a source of information on what aspects of a discussed product can be improved upon. These aspects are named drivers. CQM, the company in which the internship for which this thesis was written took place, has developed a tool in which reviews are manually annotated for a fixed set of drivers. The result is that for each driver, each review can be assigned a driver score. The driver score can be a positive or negative number, indicating with what sentiment the reviews discusses the driver, or left blank, meaning that the review does not discuss the driver. The goal of this work is to (partially) automate this process, so that, given the review, the driver scores can be predicted.

The first step towards achieving this is creating a binary classification problem for every driver, where a binary variable can, for example, indicate whether a review discusses the driver or not. A step further is multinomial classification, where one can also distinguish whether a driver is discussed with positive or negative sentiment.

The reviews are represented as variables, with every variable representing the use of a word. In this thesis, different forms for these variables are experimented with. With every variable representing a unique word, a large number of distinct predictors is offered for the classification problem. Given these predictors, two types of models are considered to solve the classification problem: Elastic net models and random forests. Both types of models need to be adapted for the class imbalance before they can be used for the classification problem of predicting the driver scores. The results of these models are evaluated using the area under the curve (AUC) and for the multinomial problem a multinomial generalisation, the AUC_μ. These measures are chosen, because they are effective at evaluating the performance of our models in the context of the class imbalance.

The results were ultimately evaluated for various variable forms and models. For the model we found that a random forest adapted to use stratified bootstrap samples to grow decision trees gave strong performance, especially when combined with variables that were given an indicator function form or normalized tf-idf form.

Keywords: document classification, topic detection, sentiment detection, imbalanced classification problems.

Preface

With this thesis, both my graduation project and student life come to an end. It feels like only yesterday that I moved to Delft and took a seat for a lecture in applied mathematics for the very first time. Nevertheless, I can look back at a wonderful and memorable journey that brought me to the point where I am now, about to finish my studies.

For the past eight months I have been working on the thesis you are now reading as a part of my graduation project at CQM in Eindhoven. I have come to know the company as a truly warm place where people work out of passion and interest. I would like to thank all the colleagues for the great time I had, not only when it came to work and research, but also for those joyful moments after working hours. Special thanks go to my mentors Jan Willem Bikker and Peter Stehouwer. Due to their sincere interest and helpfulness, every meeting was filled with new ideas about how to move forward, which always gave me fresh energy and motivation.

Furthermore, I would like to thank my supervisor Geurt Jongbloed, without whom I would probably have never ended up doing this project at CQM. Geurts critical assessment has helped bring out the best of my mathematical abilities and I always enjoyed the talks we had, regardless of whether these were about mathematics or life in general.

Then, I would like to thank my parents and sister of course, who have always supported me, not only during this graduation project, but through all the ups and downs of university and high school before that. The knowledge that they will always support me in realizing my ambitions, regardless of whether these lie in the orchards in Zeeland or the university in Delft, is something I am grateful for.

Lastly, I would like to thank all the friends that I met in so many different places and situations during this journey. Some of them I have already known since I was very young, others I met during my university life: inside or outside the lecture room, having a beer or having no beer, doing sports or already having done some sports, abroad, in Eindhoven or in Delft. Regardless of where and how I met them, I would like to thank all of them for making these years as enjoyable as they were. I now look forward to a new phase of life and I can only hope that it will be as enjoyable as the last.

Jos Steijn
Eindhoven, May 2020

Nomenclature

α	Elastic net model parameter
λ	regularization parameter
$(\widehat{p}_{i,j})_q$	estimate for $(p_{i,j})_q$
$(\widehat{p}_j)_q$	estimate for $(p_j)_q$
$(p_{i,j})_q$	$(p_j)_q$ for document i
$(p_j)_q$	probability for class q as outcome for driver j as a stochastic function
$\mathbb{T}_K$	the K-simplex
$\widehat{d}'_{i,j}$	estimate for $d'_{i,j}$
$\widehat{p}_{i,j}$	$\widehat{p}_j$ for document i
$\widehat{p}_j$	estimate for p_j
A	misclassification matrix
B	number of trees in random forest
C_j	values attained by d_j in the data set
$d'_{i,j}$	coarsened driver score d_j for document i
d'_j	coarsened driver score for driver j
$d_{i,j}$	driver score d_j for document i
d_j	driver score for driver j
i	document
j	driver
m	number of drivers
n	number of documents
P	number of classes
$p_{i,j}$	p_j for document i
p_j	probability for driver j as a stochastic function
q	class
t	threshold value
V	vocabulary size

v	word in vocabulary
$x_{i,v}$	x_v for document i
x_v	variable for word v
AUC	area under the curve
FN	false negatives
FP	false positives
FPR	false positive rate (= 1−sensitivity)
GI	Gini impurity
LASSO	least absolute shrinkage and selection operator
RF	random forest
ROC	receiver operating characteristic
TN	true negatives
TP	true positives
TPR	true positive rate (=sensitivity)

Contents

1 Introduction

The last decade,giant steps have been made in the world of data science and analytics. With the availability of big data and fast computers, better and large-scale analyses can be done. For companies, these analyses are key, as it is believed that lots of information and knowledge can be retrieved from the logged data and data that is freely available online. The field of applications of big data that this thesis focuses on is marketing intelligence. That is, using data to gain insights into customer behaviour and opinions. Even marketing strategies can be tuned based on prediction models such that the strategy is optimal for sales or product ratings. In the next section, we will dive into the specific questions CQM [1] is asked by their clients.

1.1. Annotated reviews

In recent years, companies have asked CQM to investigate what parts of products to improve upon, based on webshop reviews of these products. In response, CQM has developed an annotation tool. The annotation tool is a piece of software that requires online reviews of one product, or a group of similar products, and preset 'drivers' as input. Drivers are subjects discussed in these reviews, and can have an associated sentiment. Examples of drivers could be 'ease of use', 'durability' or 'cost quality ratio', and a review can then be positive or negative about these drivers, or can not discuss them at all. Currently, reading these reviews and annotating for the drivers is done manually.

Consider a small example of the output of this process. One data set used within this thesis is on shaver reviews. Part of this data set looks like this:

review_text	Accessoiries	Charging_Electronic_power	Design	...
Works well, shaves closely, just like Braun ...	−1.0	NA	1.0	...
My first ever electric razor, so can't compare ...	NA	NA	NA	...
Excellent shave once you get used to it. My ...	NA	1.0	NA	...
I have a 10-year old Braun 7526 that's still going ...	NA	NA	NA	...
⋮	⋮	⋮	⋮	⋱

Figure 1.1: Example of part of the data set on shavers

In the first column, the reviews are shown. All other columns represent the driver scores. Three such columns are shown here. The driver scores shown in this example are −1, 1 and NA. −1 and 1 indicate negative and positive sentiment respectively, and NA indicates that the review does not discuss the driver. There may be more possible driver scores within this and other data sets. Within the data set from Example 1.1 the

[1] CQM stands for Consultants in Quantitative Methods and is the company in which my internship took place. The department of CQM in which this thesis is written is specialized in product and process innovation. To better innovate and improve products, insights in customer opinions are required. Therefore, research is done in extracting overarching themes and opinions from large sets of online reviews, without having to read every single review.

driver scores -2 and 2, signifying very negative and very positive sentiment, and 0, showing that a driver is discussed, but in a neutral manner, are for example also used.

1.2. Research Questions

Now, the goal of this thesis is to automate the annotation process. In other words, where the annotation tool developed by CQM currently requires manual work, reading and annotating the reviews one by one, in this thesis we aim to predict the driver scores, given the reviews. We want to translate the 'free text' from a user review, to useful data that can be used to predict the driver scores. For a start, we will try to predict whether the driver score is NA (meaning that the review does not contain information relevant for the driver) or not. Or alternatively, try to predict whether the driver score is positive (meaning that the review mentions the driver in a positive manner) or not, meaning that the review either mentions the driver in a negative manner, or not at all. Vice versa for negative driver scores. As we sort the review in either of two classes, we call this a binary classification problem. Then, we will create a model for the setting where we distinguish between positive and negative sentiment, and NA. This is called the multinomial classification problem, where we sort the review in either of three classes. The latter may prove challenging, as the many subtleties in language may make it hard to distinguish between negative and positive driver scores. We do not necessarily aim to predict these driver scores correctly one hundred percent of the time. A good guess at how the review should be annotated would already save a lot of work when using the annotation tool.

Apart from the goal of what we want to predict, we also differentiate in the methods used for prediction. In this thesis, two different methods will be discussed extensively; Elastic net (including LASSO and Ridge regression) and random forests. Furthermore, we will discuss how to transform reviews to variables, and the possibilities we have in how to choose these variables. Aside from the input, we will show how to evaluate the output of these methods using the AUC and a multinomial generalization of the area under the curve (AUC). As a conclusion, the following research questions are formulated:

- Using annotated data, can driver presence and sentiment be predicted for new reviews, and if so, what is the accuracy of these predictions?
  - What challenges arise when going from a binary classification to a multinomial classification and how can these be overcome?
  - How do the Elastic net and random forest models function, how are they different, and how do we compare them?
  - How can we maximize the predictive performance of our models and critically evaluate our results?
  - To what extent can we gain insight into which words or which combinations of words in reviews are relevant for predictions?

1.3. Outline of this thesis

In Chapter 2, we will first briefly introduce the data sets used within this thesis, after which we mathematically formalize the problem statement of this research. In Chapter 3 it is shown how to process the free text from reviews to variables. In Chapter 4 we will discuss how to evaluate our results, regardless of the method used. Chapter 5 will describe the Elastic net models and in Chapter 6 Random Forests will be introduced. In Chapter 7 we will discuss how we can use the methods described in earlier Chapters to obtain results and how we can compare these results. In Chapter 8 the results will be shown and compared and we will have found the best model. Then, in Chapter 9 we will zoom in and show how our the obtained models function when applied to individual reviews. Lastly, in Chapter 10, we will give our conclusions, critical notes and further recommendations.

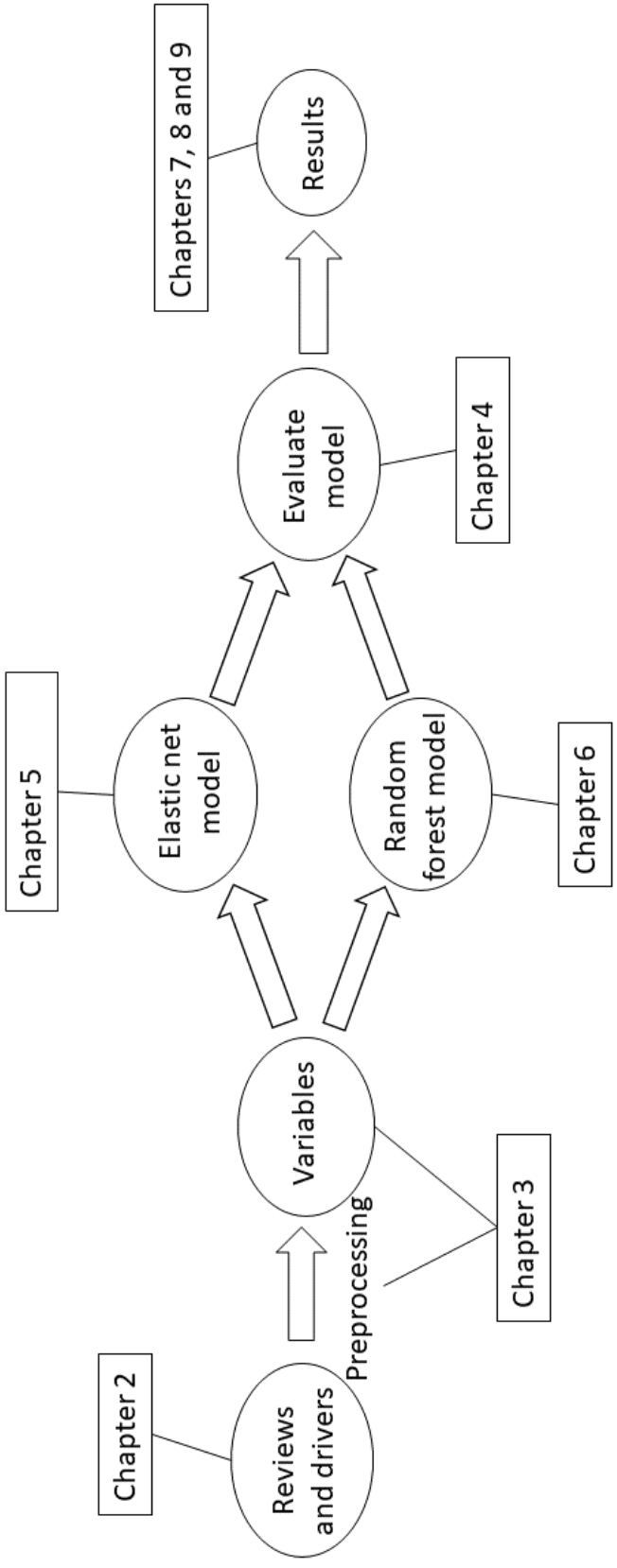


Figure 1.2: Illustrative overview of the outline of this thesis.

2

Outcome and prediction models

In the first section of this chapter, we will introduce the data sets used within this thesis, and give some information about these data sets that will be of use later on. In the rest of the chapter, we formalize the mathematical problem statement of this thesis.

2.1. Data sets description

Remember the example from the shaver data set that was introduced earlier:

review_text	Accessories	Charging_Electronic_power	Design	...
Works well, shaves closely, just like Braun ...	−1.0	NA	1.0	...
My first ever electric razor, so can't compare ...	NA	NA	NA	...
Excellent shave once you get used to it. My ...	NA	1.0	NA	...
I have a 10-year old Braun 7526 that's still going ...	NA	NA	NA	...
⋮	⋮	⋮	⋮	⋱

Figure 2.1: Example of part of the data set on shavers

In the first column, the reviews are written. All other columns represent the driver scores. In this thesis, two data sets have been used. They have been annotated for different drivers. We will discuss the properties of these data sets in this chapter. This will help us understand some of our results later on, and possibly even help us achieve better results.

Both data sets contain annotated reviews on shavers. The first set contains 1976 annotated reviews. These reviews are randomly selected user comments on shavers from various web pages. The reviews are annotated for 17 drivers. The second data set contains 775 reviews, also on shavers. These reviews have been collected in such a matter that they are uniformly distributed over the star ratings users submit as part of their review on the website (1-5 stars scale). This means that the sample is stratified, in that there are as many 1 star reviews as there are 2 star reviews as there are 3 star reviews etc. This data set is annotated for 15 drivers. We will refer to the first data set as the $n1976-$ data set and the second as $n775-$ data set.

2.1.1. Driver sentiments

The drivers in the data sets are the following:

0.	Shaving (general)	1.	Accessories
1.	During shaving – speed	2.	Charging & Electronic power
2.	During shaving – comfort	3.	Design
3.	After shaving – irritation	4.	Features
4.	After shaving – closeness	5.	Issues & Reliability
5.	Cofo	6.	Service & Delivery
6.	Sound	7.	Value for money
7.	Ease of use	8.	cleaning
8.	Cleaning	9.	ease of use
9.	Smart Clean	10.	shaving closeness
10.	Service & Delivery	11.	shaving comfort
11.	Accessories	12.	shaving irritation
12.	Features	13.	shaving rest
13.	Charging & electronic power	14.	shaving speed
14.	Design	15.	smart clean
15.	Value for money		
16.	Issues & reliability		

Figure 2.2: Drivers for the $n1976$ – data set (on the left) and for the $n775$ – data set (on the right).

We see that there are 17 drivers in the $n1976$ – data set and 15 drivers in the $n775$ – data set. Note that the first driver in the $n1976$ – data set is indexed as 0. This is because it is not really a driver in the same sense as the others, but its scoring represents the overall sentiment of the review. Nonetheless, it is regarded as being a driver in this thesis, as it still interesting to see to what extent we can predict it. We treat it the same as all other drivers.

It is important to note that there is large variance in the number of non-NA driver scores among these drivers. For example, more reviews address 'Accessories' than there are reviews that address 'Features'. Distinguishing positive (> 0), negative (< 0) and neutral ($= 0$) sentiment among the driver scores, gives the following distributions for the drivers.

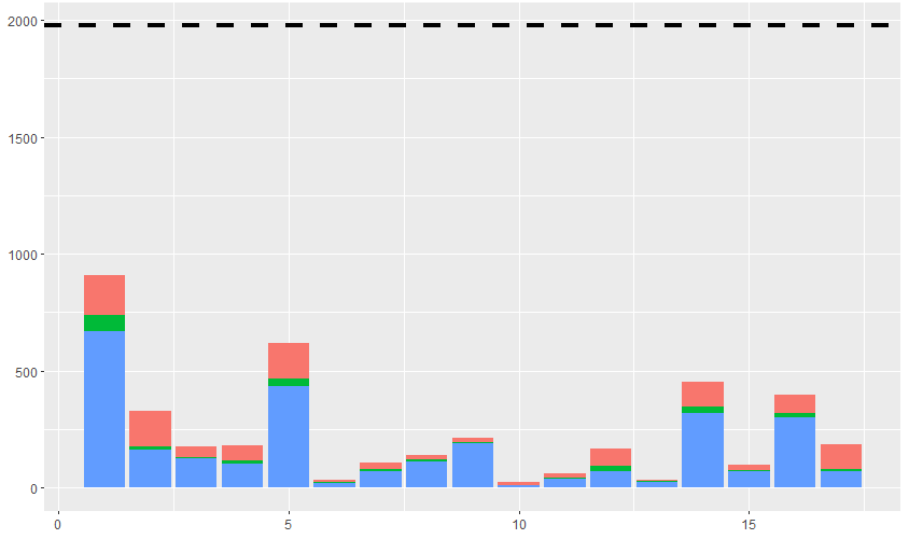


Figure 2.3: Histogram of the number of driver scores for every driver in the $n1976$ – data set. In blue the number of positive, in green the neutral and in red the negative driver scores. The horizontal dashed line gives the total number of reviews in the data set.

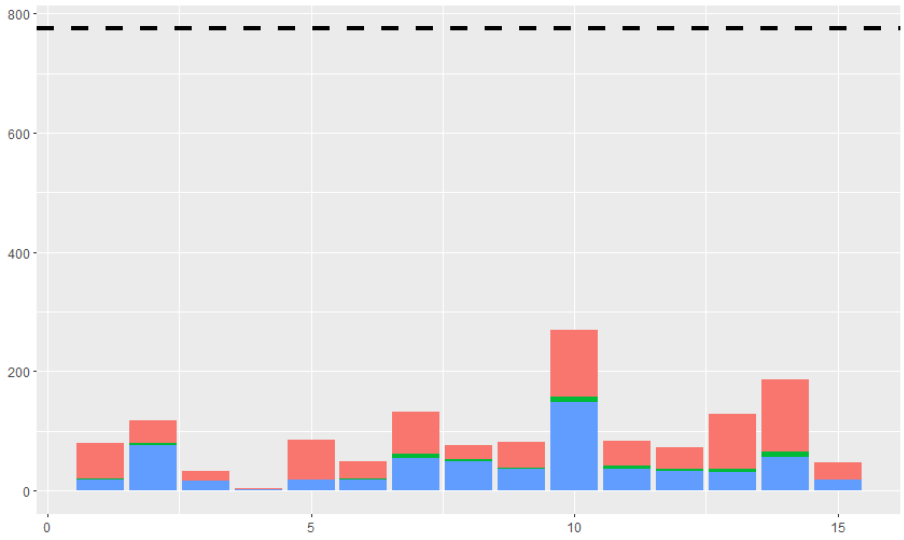


Figure 2.4: Histogram of the number of driver scores for every driver in the $n775-$ data set. In blue the number of positive, in green the neutral and in red the negative driver scores. The horizontal dashed line gives the total number of reviews in the data set.

Thus, we see that the reviews in the $n1976-$data set are predominantly positive. For the reviews in the $n775-$ data set the opposite is true; they are predominantly negative, which is most likely a result of these reviews being a sample of reviews uniformly distributed among star ratings 1-5. There are drivers which contain more/less positive/negative driver scores than other drivers, probably a result of people being more positive or negative about certain aspects of the product than about others. Furthermore we see that for every driver in both data sets, the number of reviews not mentioning the driver is larger than the number of reviews mentioning it, whatever the sentiment.

2.1.2. Review lengths

Both data sets contain reviews of varying lengths. It may be useful for later analysis to have an idea as to how long these reviews are.

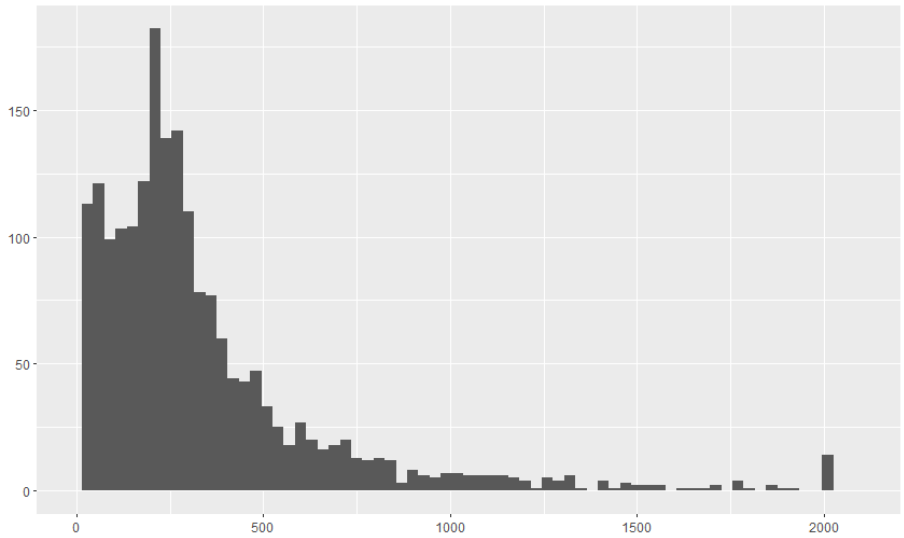


Figure 2.5: Histogram of the review lengths of the reviews in the $n1976-$ data set. Review length is the number of all characters in the review, including blank spaces. A few outliers of review length up to 6000 are added to the most right stack in the histogram.

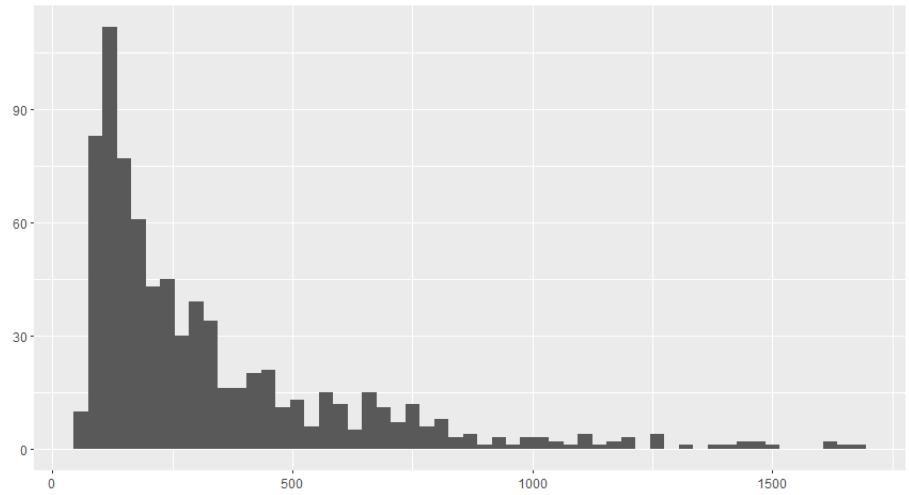


Figure 2.6: Histogram of the review lengths of the reviews in the $n775$ – data set. Review length is the number of all characters in the review, including blank spaces.

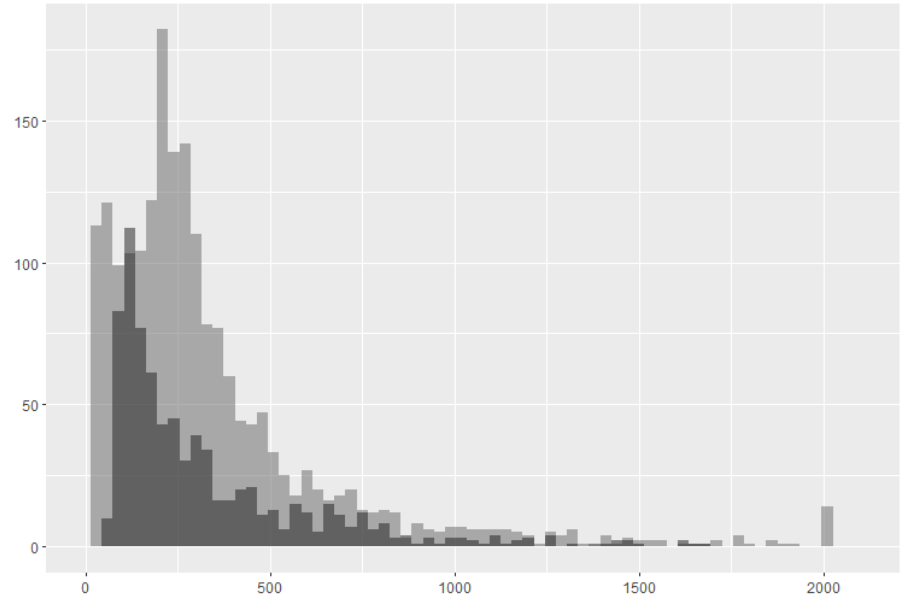


Figure 2.7: Comparison of the histograms of the review lengths of the reviews in the $n775$ – data set (dark gray) and the $n1976$ – data set (light gray). Review length is the number of all characters in the review, including blank spaces. A few outliers of review length up to 6000 in the $n1976$ – data set are added to the most right stack in the histogram.

The average length of the reviews for the $n1976$ – data set is 336 and for the $n775$ – data set 331 (rounded to whole numbers). The length of the reviews is similarly distributed among both data sets. Both contain a few outliers with long review lengths, with the $n1976$ – data set having a few extreme ones.

2.2. Modelling driver outcomes

In this section, we will introduce the mathematical notation for our problem. Remember the example from the shaver data set that was already introduced earlier:

review_text	Accessories	Charging_Electronic_power	Design	...
Works well, shaves closely, just like Braun ...	−1.0	NA	1	...
My first ever electric razor, so can't compare ...	NA	NA	NA	...
Excellent shave once you get used to it. My ...	NA	1.0	NA	...
I have a 10-year old Braun 7526 that's still going ...	NA	NA	NA	...
⋮	⋮	⋮	⋮	⋱

Figure 2.8: Example of part of the data set on shavers

The first column contains the written reviews, the other columns represent the driver scores. In the general context of this thesis, we say that we are given a set of n annotated reviews, and refer to one of these documents as $i \in \{1, ..., n\}$, and n is the total number of reviews or documents. The terms 'document' and 'review' are interchangeable within the rest of this thesis. In the rest of this chapter, we will focus on the driver outcomes. We will not yet discuss how we are going to predict them, but properly define them and discuss how to evaluate our future predictions.

2.3. Modelling driver outcomes

So, every data set used in this thesis contains drivers, and for each document, every driver has an annotated value attached. We say that we have driver $j \in \{1, ..., m\}$, in our data set, with m being the total number of drivers. We introduce d_j to denote the score given for a driver j in general. To introduce the actual score for a driver j for a document i, we introduce $d_{i,j}$. In the context of our example, we thus see that $d_{1,1} = -1.0$, $d_{1,2} = \text{NA}$ and $d_{1,3} = 1.0$.

We will now introduce new variables $d'_{i,j}$, which we will try to predict. The outcomes of $d'_{i,j}$ depend directly $d_{i,j}$. We will discuss how in the rest of this section.

2.3.1. Binary outcomes

For a start, we transform the driver scores $d_{i,j}$, and the general form d_j, to binary values. Why this? In all data sets used in this thesis, each driver j attains values in a discrete set of values. Predicting all values exactly will prove challenging, due to a lack of data. Thus, we split the set of values attained in two, and create the situation in which our goal is to predict binary data. In this way, we 'coarsen' or 'bin' the data. We already looked at one coarsening of the data: in Figure 2.3 and Figure 2.4 we coarsened the data to 4 levels; positive sentiment (driver score > 0), negative sentiment (driver score < 0) and neutral sentiment (driver score = 0), and, although not really shown in the picture, a review not mentioning the driver (driver score = NA). This is a multinomial coarsening. For now we will look at a binary division.

Note that, in the context of this thesis, it may be desirable to look at multiple ways of coarsening the data. For example, we may look at predicting whether a driver is discussed or not, which would mean predicting a value of NA versus any other value. Or, we might only be interested in knowing when the driver is discussed with positive sentiment, which would come down to predicting whether a value greater than 0 is obtained, or not, regardless of whether it is a negative, 0 or NA in that case. Formally, we write the following:

Take a driver j. We denote the set of values j attains in the data with C_j, and split C_j in two disjoint sets; one set being C_j^1, and the other C_j^0. Now, for a fixed $i \in \{1, ..., n\}$ and fixed $j \in \{1, ..., m\}$, a new binary variable d'_j is defined as $d'_{i,j} := 1$ if $d_{i,j} \in C_j^1$ and $d'_{i,j} := 0$ if $d_{i,j} \in C_j^0$. To sum, this up we get the following formulation:

$$d_{i,j} \in C_j, \qquad \forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, m\} \tag{2.1}$$

$$C_j = C_j^1 \cup C_j^0, \quad C_j^1 \cap C_j^0 = \emptyset, \qquad \forall j \in \{1, \dots, m\} \tag{2.2}$$

$$d_{i,j} \in C_j^1 \iff d'_{i,j} = 1, \quad d_{i,j} \in C_j^0 \iff d'_{i,j} = 0 \qquad \forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, m\} \tag{2.3}$$

We name $d'_{i,j}$ the *coarsened* or *binned* driver scores. Colloquially, we will also sometimes simply refer to

them as driver scores, when it is clear that we do not mean $d_{i,j}$. In similar fashion, we define d'_j as the transformation of the general form d_j, taking only values in $\{0,1\}$ in the binary case.

In this thesis we define three types of choices for C_j^1 and C_j^0, which are the ones we will mainly use. We refer to these as different types of *sentiment*, and let them be defined as

1. Present: $C_j^1 = \{c \in C_j : c \neq NA\}$, $C_j^0 = \{c = \text{NA}\}$
2. Positive: $C_j^1 = \{c \in C_j : c \geq 0\}$, $C_j^0 = \{c \in C_j : c < 0 \vee c = \text{NA}\}$
3. Negative: $C_j^1 = \{c \in C_j : c < 0\}$, $C_j^0 = \{c \in C_j : c \geq 0 \vee c = \text{NA}\}$

Present can be thought of as detecting whether a review mentions a driver or not. It is therefore not really sentiment, but we will still refer to it as 'present sentiment' in the rest of this thesis. Positive sentiment refers to detecting whether a driver scores is greater than or equal to zero, and negative to detecting whether a driver score is smaller than 0. $d_{i,j} = 0$ is rare within both data sets (see Figures 2.3 and 2.4), but has been added to positive sentiment for convenience. Many more choices of C_j^1 and C_j^0 can be thought of, but these are the choices that will mainly be used within this thesis.

In the rest of this thesis, it will be useful to look at the driver scores for multiple documents at the same time. Say that we have a subset of documents $S \subseteq \{1,\dots n\}$, then we define

$$d'_{S,j} := \begin{bmatrix} d'_{1,j} \\ \vdots \\ d'_{n_s,j} \end{bmatrix} \tag{2.4}$$

Where n_s is the size of the set S. To prevent any confusion: the first element of this vector is not necessarily also the first element of the data set. Furthermore, we will use this notation later in this thesis also for cases where S is not a subset of of $\{1,\dots n\}$.

2.3.2. Multinomial outcomes

As mentioned earlier, in the data every driver d_j takes its values in a discrete set that we name C_j. In equation 2.2, we split C_j in two discrete sets C_j^1 and C_j^0, creating binary variables $d'_{i,j}$ as in equation 2.3. Now we will generalize these formulations and create multinomial outcomes for $d'_{i,j}$. Consider not dividing the outcomes in two classes, but in $P > 2$ classes. We then have that

$$d_{i,j} \in C_j, \qquad \forall i \in \{1,\dots,n\}, \forall j \in \{1,\dots,m\} \tag{2.5}$$

$$C_j = \bigcup_{p=1}^{P} C_j^p, \qquad \forall j \in \{1,\dots,m\} \tag{2.6}$$

$$C_j^p \cap C_j^q = \emptyset, \qquad \forall p,q = \{1,\dots,P\}, p \neq q, \forall j \in \{1,\dots,m\} \tag{2.7}$$

In a binary case, the possible outcomes for $d'_{i,j}$ are two classes 0 and 1. In the multivariate case, we now extend this to P classes. We say that:

$$d_{i,j} \in C_j^p \iff d'_{i,j} = p, \quad d_{i,j} \in C_j^q \iff d'_{i,j} = q \qquad \forall p,q = \{1,\dots,P\}, \forall i \in \{1,\dots,n\}, \forall j \in \{1,\dots,m\} \tag{2.8}$$

In similar fashion, we define d'_j as the transformation of the general form d_j, taking values in $\{1,...,P\}$ in the multinomial case. In this thesis we mainly use one partition of C_j in the multinomial setting. In this one, we define classes $\{P,N,\text{NA}\}$ with

1. $C_j^P = \{c \in C_j : c \geq 0, c \neq \text{NA}\}$

2. $C_j^N = \{c \in C_j : c < 0, c \neq \text{NA}\}$

3. $C_j^{\text{NA}} = \{c \in C_j : c = \text{NA}\}$

Many other partitions of C_j can be thought of, but this is the one that we will work with in the multinomial setting.

2.4. Probabilistic model

2.4.1. Binary model

In this thesis, we assume that the outcome $d_{i,j}$, and thus by the above also $d'_{i,j}$, is generated by a stochastic function with document i as its input. In the binary setting, this simply means that $d'_{i,j} = 1$ with a certain probability based on document i, and $d'_{i,j} = 0$ with one minus that probability. We name the stochastic function giving this probability p_j, so that we have a different function for every driver j.

In the binary setting, this gives rise to the following model:

$$p_j(\text{doc}) := P\left(d'_j = 1|\text{doc}\right), \qquad \forall j \in \{1,\dots,m\} \tag{2.9}$$

$$d'_j|\text{doc} \sim \text{Bern}\left(p_j(\text{doc})\right), \qquad \forall j \in \{1,\dots,m\} \tag{2.10}$$

where Bern is short for Bernoulli distribution and doc represents all information obtained from a review or document. It immediately follows that

$$1 - p_j(\text{doc}) := P\left(d'_j = 0|\text{doc}\right), \qquad \forall j \in \{1,\dots,m\} \tag{2.11}$$

So for a specific document i we have

$$p_j(\text{doc}_i) = P\left(d'_{i,j} = 1|\text{doc}_i\right), \qquad \forall i \in \{1,\dots,n\}, \forall j \in \{1,\dots,m\} \tag{2.12}$$

We define the notation

$$p_{i,j} := p_j(\text{doc}_i) \tag{2.13}$$

for use later on. Furthermore, we say that $d'_{i,j}|\text{doc}_i$ is a draw from the distribution $\text{Bern}\left(p_{i,j}\right)$.

2.4.2. Multinomial model

In the multinomial case we have P possible outcomes for $d'_{i,j}$ instead of just two. Thus, we generalize the probabilistic model and obtain:

$$\left(p_j\right)_q(\text{doc}) := P\left(d'_j = q|\text{doc}\right), \qquad \forall q \in \{1,\dots,P\}, \forall j \in \{1,\dots,m\} \tag{2.14}$$

$$d'_j|\text{doc} \sim \text{Cat}\left(\left(p_j\right)_1(\text{doc}),\dots,\left(p_j\right)_P(\text{doc})\right), \qquad \forall q \in \{1,\dots,P\}, \forall j \in \{1,\dots,m\} \tag{2.15}$$

where Cat is short for a categorical distribution. We also define

$$p_j = \left[\left(p_j\right)_1,\dots,\left(p_j\right)_P\right], \qquad \forall j \in \{1,\dots,m\} \tag{2.16}$$

The categorical distribution is a special case of the multinomial distribution where $n = 1$. [1] Additionally, we have the constraint that

$$\sum_{q=1}^{P}\left(p_j\right)_q = 1, \qquad \forall q \in \{1,\dots,P\}, \forall j \in \{1,\dots,m\} \tag{2.17}$$

Now, we introduce the *simplex*.

[1] The multinomial distribution is generally specified for parameters $n \in \mathbb{N}$ and $p_1,\dots,p_k$ such that $\sum_{i=1}^{k} p_i = 1$, where k is the number of classes and n the number of trials.

Definition 2.1 (K-simplex)
The K-simplex $\mathbb{T}_K$ in $\mathbb{R}^{K+1}$ is the set

$$\mathbb{T}_K = \left\{ \left[x_1, \ldots, x_{K+1} \right] : x_k > 0,\ 1 \leq k \leq K+1,\ \sum_{k=1}^{K+1} x_k = 1 \right\}$$

As an example, we illustrate $\mathbb{T}_2$.

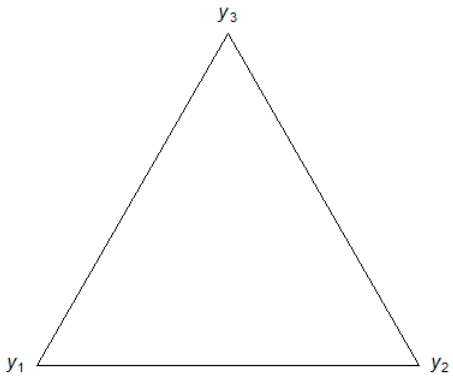


Figure 2.9: $\mathbb{T}_2$, the 2-simplex. Corners of the simplex are given by $y_1 = \left[1,0,0\right]$, $y_2 = \left[0,1,0\right]$ and $y_3 = \left[0,0,1\right]$.

Note that $\mathbb{T}_2$ is a subset of $\mathbb{R}^3$, but due to the restriction that the entries of every element in the set must sum to 1, we can depict $\mathbb{T}_2$ in two dimensions. When the first two entries of a vector in $\mathbb{T}_2$ are known, the third entry is known as well. This is why $\mathbb{T}_K$ is an element of $\mathbb{R}^{K+1}$: we only need the first K entries of a $(K+1)$-dimensional vector, to know the last element of the vector.

Thus, we see that p_j is in the $(P-1)$-simplex, i.e. $p_j \in \mathbb{T}_{(P-1)}$. Now, given a document i we have that

$$\left(p_j\right)_q (\text{doc}_i) = P\left(d'_{i,j} = q | \text{doc}_i\right), \qquad \forall q \in \{1,\ldots,P\}, \forall i \in \{1,\ldots,n\}, \forall j \in \{1,\ldots,m\} \tag{2.18}$$

We define the notation

$$\left(p_{i,j}\right)_q := \left(p_j\right)_q (\text{doc}_i) \tag{2.19}$$

for use later on and we say that $p_{i,j}$ is an element of $\mathbb{T}_{(P-1)}$ given by

$$p_{i,j} := \left[\left(p_{i,j}\right)_1, \ldots, \left(p_{i,j}\right)_P \right] \tag{2.20}$$

Furthermore, we say that $d'_{i,j}|\text{doc}_i$ is a draw from the distribution $\text{Cat}\left(\left(p_j\right)_1(\text{doc}_i), \ldots, \left(p_j\right)_P(\text{doc}_i)\right)$.

2.5. Estimating probabilities

Now, the functions p_j are generally not known. Thus, we will estimate p_j in this thesis. We name these estimates $\widehat{p}_j$, based on the couples $\left(\text{doc}_1, d_{1,j}\right), \ldots, \left(\text{doc}_n, d_{n,j}\right)$.

In the multinomial setting the same procedure is followed and $\widehat{p}_j$ becomes a vector of length P with entries $\left(\widehat{p}_j\right)_q$ where $q \in \{1,\ldots,P\}$, such that $\sum_{q=1}^{P} \left(\widehat{p}_j\right)_q = 1$. The entries $\left(\widehat{p}_j\right)_q$ are estimated individually.

This finishes the introduction of the notation for estimation of the model. Using this notation, we can move towards the prediction of driver values for future documents.

2.6. Predictions

Say that we have n annotated documents, meaning that we have the driver scores $d_{i,j}$, and can thus derive the coarsened driver scores $d'_{i,j}$. Now, a new review, say $i = n+1$, comes in. We do not have the driver score $d_{n+1,j}$ and we want to predict it. Then, we can derive $\widehat{p}_j$, as introduced in the previous section, based on the n observations we have.

Using $\widehat{p}_j$, we will then try to predict the driver scores for the new document. We define

$$\widehat{p}_{n+1,j} := \widehat{p}_j(\text{doc}_{n+1}), \qquad \forall\, j \in \{1,\dots,m\} \tag{2.21}$$

2.6.1. Binomial predictions

Now, we will make a prediction $\widehat{d}_{n+1,j}$ for $d_{n+1,j}$. We will first introduce the concept of a threshold value.

Definition 2.2 (threshold value)
Given a binary variable d for which we want to make a prediction $\widehat{d}$ and are given a probability $p \in [0,1]$, we say that $t \in [0,1]$ is a threshold value when the prediction is chosen as

$$\widehat{d} := \begin{cases} 1, & p \geq t \\ 0, & p < t \end{cases}$$

We say that $\widehat{d}$ is the prediction found by applying the threshold value t to p. In the context of this thesis, this means that we predict $\widehat{d}_{n+1,j} = 1$ for a threshold value t when $\widehat{p}_{n+1,j} \geq t$ and $\widehat{d}_{n+1,j} = 0$ when $\widehat{p}_{n+1,j} < t$. A logical choice would be to choose $t = 0.5$. Later in this thesis we will see that in some situations other choices for t can prove very useful. Implicitly, this means that

$$\widehat{p}_j\left(\text{doc}\right) = P\left(\widehat{d}'_j = 1 | \text{doc}\right) \tag{2.22}$$

and

$$\widehat{p}_{i,j} = \widehat{p}_j\left(\text{doc}_i\right) = P\left(\widehat{d}'_{i,j} = 1 | \text{doc}_i\right) \tag{2.23}$$

We will often not be interested in predicting just one new document, but in predicting a set of new documents, say $n+1,\dots,n+n_{\text{new}}$. For future purposes, we define the notation for these sets, their estimated probabilities and predictions here. We define

$$I_{\text{new}} := \{n+1,\dots,n+n_{new}\} \tag{2.24}$$

$$\widehat{p}_{I_{\text{new}},j} := \begin{bmatrix} \widehat{p}_{n+1,j} \\ \vdots \\ \widehat{p}_{n+n_{\text{new}},j} \end{bmatrix}, \qquad \forall\, j \in \{1,\dots,m\} \tag{2.25}$$

$$\widehat{d}'_{I_{\text{new}},j} := \begin{bmatrix} \widehat{d}'_{n+1,j} \\ \vdots \\ \widehat{d}'_{n+n_{\text{new}},j} \end{bmatrix}, \qquad \forall\, j \in \{1,\dots,m\} \tag{2.26}$$

and we have that

$$|I_{new}| = n_{\text{new}} \tag{2.27}$$

$$\widehat{p}_{I_{\text{new}},j} \in [0,1]^{n_{\text{new}} \times 1} \tag{2.28}$$

$$\widehat{d}'_{I_{\text{new}},j} \in \{0,1\}^{n_{\text{new}}} \tag{2.29}$$

2.6.2. Multinomial predictions

Next, we will define our predictions for the multinomial setting. In the multinomial setting the outcome $d'_{i,j}$ is a draw from the categorical distribution with parameters $p_{i,j} := \left[\left(p_{i,j}\right)_1, \ldots, \left(p_{i,j}\right)_P\right]$. $p_{i,j}$ must be in the $(P-1)$-simplex, i.e. $p_{i,j} \in \mathbb{T}_{P-1}$. The vertices in the $(P-1)$-simplex correspond to the classes $\{1,\ldots,P\}$. We will split $\mathbb{T}_{P-1}$ in P parts, and predict a certain class when the estimate $\widehat{p}_{i,j}$ falls in the part of the corresponding class. To this end, we first introduce the *partition matrix*.

Definition 2.3 (Partition matrix)
Let us have a classification problem with P classes, and let two classes be $q_1, q_2 \in \{1,\ldots,P\}$. Let A be a $P \times P$ matrix and let A_{q_1,q_2} be the cost of classifying an instance as class q_1 when its true class is q_2. Then A is a partition matrix and defines a partition on the $(P-1)$-simplex.

The partition matrix A is to the multinomial setting what the threshold value t is to the binary setting. It can be shown that the threshold value can be derived from a 2×2 partition matrix. Furthermore, it can be shown that any partition matrix A can be expressed as some matrix A' with $A'_{q,q} = 0, \forall q \in \{1,\ldots,P\}$ and $A'_{q_1,q_2} \neq 0, \forall q_1 \neq q_2$.[11] A matrix of this form is also referred to as a *misclassification matrix*. We will use both terms within this thesis. We say that the partition matrix A induces *decision boundaries* between the P classes.

Definition 2.4 (Decision boundary)
Let us have a classification problem with P classes with partition matrix A, and let two classes be $q_1, q_2 \in \{1,\ldots,P\}, q_1 \neq q_2$. A decision boundary between class q_1 and class q_2, $q_1 \neq q_2$, is the hyperplane that separates the two classes in the $(P-1)$-simplex. Let p be an element of the $(P-1)$-simplex and let $p_q, q \in \{1,\ldots,P\}$, be entries of p. We can find the decision boundary by using the partition matrix to solve the hyperplane of solutions that have the same losses with respect to the cost when assigned to class q_1 as they do when assigned to class q_2. The hyperplane is given by the solutions p for

$$\sum_{q=1}^{P} A_{q_1,q} p_q = \sum_{q=1}^{P} A_{q_2,q} p_q \tag{2.30}$$

for $q_1, q_2 \in \{1,\ldots,P\}$ and $q_1 \neq q_2$.

Using the dot product and $A_{k,\cdot}$ to refer to the kth row in the matrix A, we can also write the formula as

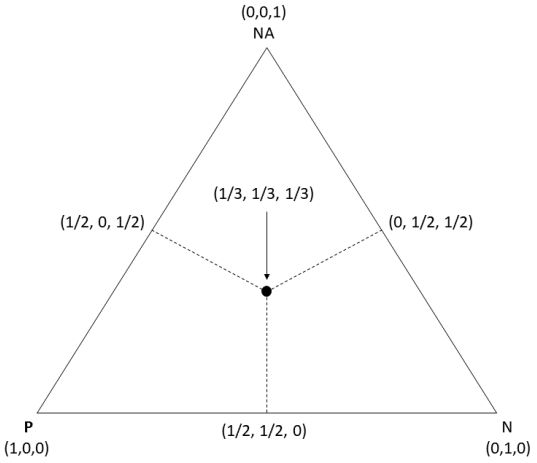
$$A_{q_1,\cdot} \cdot p = A_{q_2,\cdot} \cdot p \tag{2.31}$$

A partition matrix has at least one *equal risk point*, where the losses with respect to assigning a certain class are the same for all classes[11]. In the $(P-1)$-simplex, the equal risk point is shown as the intersection of the decision boundaries. In this thesis, we will work with 3 different choices for the partition matrix A. We will introduce these first, and show the decision boundaries they induce between the P classes that divide the $(P-1)$-simplex.

In Subsection 2.3.2 we introduced the partition of C_j used within this thesis for the multinomial setting. Using this partition, we have 3 classes, and we will thus let A be a matrix of size 3×3. A relatively simple choice for A is given in Figure 2.10.

prediction(↓), true value (→)	P	N	NA
P	0	1	1
N	1	0	1
NA	1	1	0

(a) Partition matrix A.



(b) Partition of the 2-simplex.

Figure 2.10: Example of a partition matrix A and the corresponding partition of the 2-simplex.

In the example in Figure 2.10a, equal cost is given to every misclassification. There is no difference in classifying a true P as N or NA or vice versa; every misclassification costs 1. Furthermore, Figure 2.10b shows the corresponding partition of the 2-simplex. The vertices correspond to the classes. The lines between the 3 parts of the 2-simplex are parts of the decision boundaries induces the partition matrix as in Figure 2.10a. Parts of the decision boundaries, because a decision boundary between two classes as introduced in Definition 2.4 extend further than the equal risk point (where the decision boundaries meet). Two other choices for A are introduced at the end of this section.

Now to come to our prediction. Let $\left(\mathbb{T}_{P-1}\right)_q$ be the part of $\mathbb{T}_{P-1}$ for which, when $\widehat{p}_{i,j}$ is in it, we set $\widehat{d}'_{i,j} = q$ as our prediction. Then we have that

$$\left(\mathbb{T}_{P-1}\right)_q = \left\{p \in \mathbb{T}_{P-1} : A_{q,\cdot} \cdot p \leq A_{q',\cdot} \cdot p, \forall\, q' \in \{1,\ldots,P\}\right\} \tag{2.32}$$

So that $\bigcup_{q=1}^{P}\left(\mathbb{T}_{P-1}\right)_q = \mathbb{T}_{P-1}$ and the intersection of $\left(\mathbb{T}_{P-1}\right)_q$ and $\left(\mathbb{T}_{P-1}\right)_p$ is given only by the boundary they share, $\forall\, p, q \in \{1,\ldots,P\}$. We then define our estimate for $d'_{n+1,j}$ as:

$$\widehat{d}'_{n+1,j} := \begin{cases} 1, & \left(\widehat{p}_{n+1,j}\right)_1 \in \left(\mathbb{T}_{P-1}\right)_1 \\ 2, & \left(\widehat{p}_{n+1,j}\right)_2 \in \left(\mathbb{T}_{P-1}\right)_2 \\ \vdots & \\ P, & \left(\widehat{p}_{n+1,j}\right)_P \in \left(\mathbb{T}_{P-1}\right)_P \end{cases} \tag{2.33}$$

Note that this definition also works for the binary setting. We then have that $p_{i,j} \in \mathbb{T}_1$, where $\mathbb{T}_1$ is simply a line split in two parts by the threshold value t. In section Chapter 4 we will discuss how we split up the simplex for higher dimensions.

Similar to how in the binary case we have equation 2.22, the above means implicitly that

$$\left(\widehat{p}_j\right)_q(\text{doc}) = P\left(\widehat{d}'_j = q|\text{doc}\right) \tag{2.34}$$

and that

$$\left(\widehat{p}_{i,j}\right)_q = \left(\widehat{p}_j\right)_q(\text{doc}_i) = P\left(\widehat{d}'_{i,j} = q|\text{doc}_i\right) \tag{2.35}$$

We also introduce notation for predicting driver scores for a set of n_{new} new documents. We have

$$I_{\text{new}} := \{n+1, \ldots, n+n_{\text{new}}\} \tag{2.36}$$

$$\widehat{p}_{I_{\text{new}},j} := \begin{bmatrix} \left(\widehat{p}_{n+1,j}\right)_1 & \cdots & \left(\widehat{p}_{n+1,j}\right)_P \\ \vdots & \vdots & \vdots \\ \left(\widehat{p}_{n+n_{\text{new}},j}\right)_1 & \cdots & \left(\widehat{p}_{n+n_{\text{new}},j}\right)_P \end{bmatrix}, \qquad \forall j \in \{1, \ldots, m\} \tag{2.37}$$

$$\widehat{d}'_{I_{\text{new}},j} := \begin{bmatrix} \widehat{d}'_{n+1,j} \\ \vdots \\ \widehat{d}'_{n+n_{\text{new}},j} \end{bmatrix}, \qquad \forall j \in \{1, \ldots, m\} \tag{2.38}$$

and we see that in the multinomial setting we have

$$|I_{\text{new}}| = n_{\text{new}} \tag{2.39}$$

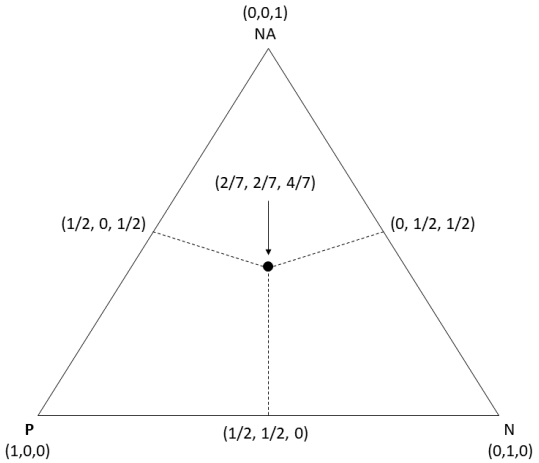
$$\widehat{p}_{I_{\text{new}},j} \in [0,1]^{n_{\text{new}} \times P} \tag{2.40}$$

$$\widehat{d}'_{I_{\text{new}},j} \in \{0,1\}^{n_{\text{new}}} \tag{2.41}$$

In Figure 2.10 we introduced one choice for the partition matrix A and showed the induced partition of the 2-simplex. This is a rather simple choice for A. We will now introduce 2 more possible choices. Within the context of user review classification as in this thesis, our first priority is identifying whether a review discusses a driver or not, and next, whether we can distinguish positive from negative sentiment. Therefore, we choose to give a higher cost to classifying a true N or P as NA or vice versa. An example of a choice of A exemplifying this and the corresponding split of the 2-simplex are given in Figure 2.11.

prediction(↓), true value (→)	P	N	NA
P	0	3/5	6/5
N	3/5	0	6/5
NA	6/5	6/5	0

(a) Partition matrix A.



(b) Partition of the 2-simplex.

Figure 2.11: Example of a partition matrix A, where higher costs have been assigned to misclassifying a P or N as NA or vice versa, and the corresponding partition of the 2-simplex.

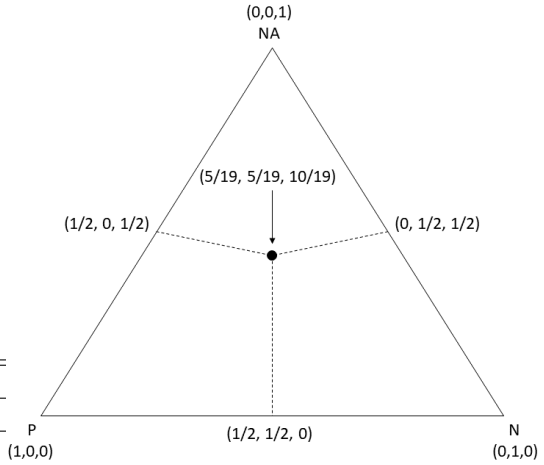
The misclassification matrix in Figure 2.11a has been chosen such that the cost of misclassifying an N or P as NA or vice versa is double that of misclassifying an N as P or vice versa. Meanwhile, the matrix has been normalized such that the sum of all entries in the misclassification matrix stays equal to the sum of all entries of the misclassification matrix in Figure 2.11a. This, so that we do not change the penalty of misclassifying in general, but only make it relative to which misclassification it concerns. The corresponding partition of the 2-simplex is given in Figure 2.11b. The change of A has altered the two of the three decision boundaries (the decision boundary between the vertices for classes P and N remains unchanged) and the location of the equal risk point has moved towards the vertice corresponding to the NA class. Note that the decision boundaries still start in the same point on the exterior of the 2-simplex. This is caused by the symmetry of A.

For example, classifying a true P as NA still costs the same as classifying a true NA as P, and this is true for all combinations of classes. Only the way we split up the interior of the 2-simplex has changed.

It is hard to judge whether the increase in cost from 1 to 2 is enough to justify our preference. Therefore we a third choice for A one could consider is shown in Figure 2.12.

prediction(↓), true value (→)	P	N	NA
P	0	3/11	15/11
N	3/11	0	15/11
NA	15/11	15/11	0

(a) Partition matrix A.



(b) Partition of the 2-simplex.

Figure 2.12: Example of a partition matrix A, where higher costs have been assigned to misclassifying a P or N as NA or vice versa, and the corresponding partition of the 2-simplex.

In the matrix in Figure 2.12, the difference in costs in misclassification has been further increased, to the point where the NA misclassifications cost 5 times as much as the misclassifications between the P and N classes.

3

From reviews to variables

We have already introduced our data sets and the corresponding mathematical notation for the driver scores and mathematical model. In this chapter, we will discuss how go to from the free text in reviews, to variables for our models. Remember the example of the raw data introduced in Example 1.1:

review_text	Accessories	Charging_Electronic_power	Design	...
Works well, shaves closely, just like Braun ...	−1.0	NA	1.0	...
My first ever electric razor, so can't compare ...	NA	NA	NA	...
Excellent shave once you get used to it. My ...	NA	1.0	NA	...
I have a 10-year old Braun 7526 that's still going ...	NA	NA	NA	...
⋮	⋮	⋮	⋮	⋱

Figure 3.1: Example of part of the data set on shavers

In this chapter we will focus on the first column shown here under `review_text` and transform the free text to variables.

3.1. From reviews to tokens

At the end of this chapter, will have multiple options for how to shape the variables obtained from the reviews. Regardless of which choice we make, we pass all reviews through a process that we call *preprocessing*, which is described in the rest of this section.

3.1.1. Bag-of-words model

In this thesis we apply a so called *bag-of-words model* to turn the reviews into data that can be used in a model. This simply means that instead of looking at the review as a whole, we look at the collection consisting of all words used in the review, the bag-of-words. So for the review "Works well, shaves closely, just like Braun ..." we find the set of words {"Works", "well", "shaves", "closely", "just", "like", "Braun",...}. Note that any punctuation and word order is thus disregarded in the bag-of-words model. The process of turning a text into a bag-of-words is also named *tokenization*, and the resulting words in the set are also called *tokens*. The terms 'tokens' and 'words' will be used interchangeably within this thesis.

3.1.2. Removing stop words

Now that we have the bag-of-words, we remove any stop words found. Stop words are generally the most common English words. Research has been done before on removing stop words in the field of sentiment

analysis and sentiment classification, and results have been inconclusive; both increase and decrease in performance has been observed[5][8]. We choose to remove them, as this is common for text analysis in general, and besides potentially improving accuracy, removing stop words has its benefits regarding computational cost. Stop words can be simple words like "I", "you" and "he", but also "why", "where" and "when". A full list of stop words is provided in Appendix A.

3.1.3. Stemming

Stemming is the process of reducing words to their word stem, base or root form. We do this so that certain groups of words in the bag-of-words are merged so that they become the same word. This is useful, because in advance we expect these words to have the same informative value for our prediction methods. An example of stemming is reducing all of the words 'works', 'working' and 'worker' to the stem 'work', and thus having all the same meaning within the bag-of-words. The specific type of stemmer we use is named a *Snowball* stemmer.[12]

The process of applying the bag-of-words model, removing stop words and stemming is referred to as the preprocessing of the data.

3.1.4. Notation

Now, we can apply the bag-of-words model, removing stop words and stemming to every review i in our data set. Name the resulting sets of words, or tokens, W_i, and name the elements $w_{i,l} \in W_i$, where $l \in \{1,...,k_i\}$ so that k_i is the size of the set W_i. We still refer to the elements of $w_{i,l}$ as words or tokens, despite the removal of stop words and stemming.

Note that it is thus very well possible that $w_{i,l_1} = w_{i,l_2}$ for $l_1, l_2 \in \{1,...,k_i\}$, and also that $w_{i_1,l_1} = w_{i_2,l_2}$ for $i_1, i_2 \in \{1,...,n\}$, $l_1 \in \{1,...,k_{i_1}\}, l_2 \in \{1,...,k_{i_2}\}$.

3.2. Matrix forms

3.2.1. Predictors $x_1, \ldots, x_V$

The question still looms: how will we predict $d'_{i,j}$ based on the bag-of-words W_i ? Given the bags-of-words, there is still no way to compare the information from one review with another. For a start, we will simply count how many times each word is used in the document. We define the vocabulary of the entire data set as the set containing the unique values among the elements of all sets W_i, thus for all $i \in \{1,\ldots,n\}$. Let V be the size of this set. We create variables $x_1,\ldots,x_V$, one variable for each word in our vocabulary.

Next, we define the $x_{i,1},\ldots,x_{i,V}$ to be the outcomes of $x_1,\ldots,x_V$ for a document i. An outcome $x_{i,v}$, $v \in \{1,...,V\}$, for document D_i, is defined to be 0 if the word associated with x_v is not in the set W_i. $x_{i,v} = 1$ if x_v occurs once in W_i, $x_{i,v} = 2$ if it occurs twice in W_i etc. We say that $x_{i,v} = \alpha$ if and only if the word associated with x_v occurs an α number of times in W_i.

3.2.2. Document term matrix

Thus, our objective is now to predict an outcome $d'_{i,j}$ based on the outcomes $x_{i,1},...,x_{i,V}$. Representing this again in a matrix form, we get:

binary_indicator	blunt	close	garbag	good	im	product	shave	...
0	1	1	1	1	1	1	1	...
1	0	0	0	0	0	0	3	...
0	0	0	1	0	0	0	0	...
0	0	0	0	0	0	0	3	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Figure 3.2: Example of the document-term matrix.

In literature, the matrix obtained by taking all columns starting from the second column of this matrix is called the *document-term matrix*. We added the first column to this. In this matrix, the left column represents our outcomes $d'_{i,j}$, which in the binary case only takes values in $\{0,1\}$, and has more possible classes in the multivariate case. The other columns in this matrix are the predictors $x_1, ..., x_V$, which take a discrete value ≥ 0.

Note that this matrix, where the left column contains data $d'_{i,j}$, is generated for one driver j. When changing to the multivariate setting, only the leftmost column changes. The rest of the matrix stays the same. Also changing the definition of $d'_{i,j}$, meaning changing the collections c_j^p from 2.2 or 2.7, only changes the first column, and has no effect on the rest of the matrix. The same goes for changing to an entirely different driver than j. Changing the driver, or coarsened driver scores $d'_{i,j}$ or even changing from the binary to the multinomial setting, thus only affects the first column of our document-term matrix. The other columns, representing predictors $x_1, \ldots, x_V$, remain the same when changing any of the above.

That does not mean that we cannot take different choices for variables $x_1, \ldots, x_V$. In the previous part of this section, the predictor variables $x_{i,1}, ..., x_{i,V}$ were introduced as representing the number of times a word appeared in a document i. i contains words $w_{i,l}$ for $l \in \{1, ..., k_i\}$. A natural augmentation is to divide the word counts $x_{i,v}$ by k_i, the total number of words in a review. We obtain a matrix of the following form:

binary_indicator	blunt	close	garbag	good	im	product	shave	...
0	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	...
1	0	0	0	0	0	0	$\frac{3}{17}$	...
0	0	0	$\frac{1}{19}$	0	0	0	0	...
0	0	0	0	0	0	0	$\frac{3}{26}$	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Figure 3.3: Example of a possible mean form matrix.

We name this matrix variant the *mean form* of the document-term matrix. This form is a natural variation from the earlier form, because in this form, the use of a word in a short review adds more value than the use of the same word in a relatively long review.

Another interesting option could be an *indicator form* document-term matrix. In the standard document-term matrix and the mean form document-term matrix we use the number of times a word is used in a document, which seems like a logical choice. In the indicator form, we abandon this idea, and simply denote a 1 when a word is used, and a 0 when it is not.

binary_indicator	blunt	close	garbag	good	im	product	shave	...
0	1	1	1	1	1	1	1	...
1	0	0	0	0	0	0	1	...
0	0	0	1	0	0	0	0	...
0	0	0	0	0	0	0	1	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Figure 3.4: Example of a possible indicator form matrix.

Thus we set $x_{i,v} = 1$ if the word x_v is used in review i, and $x_{i,v} = 0$ if x_v is not used in i. This simple form, and future results generated using this form, will put to the test whether counting the number of times a word is used in review i really gives us more information when trying to predict $d'_{i,j}$, than just knowing whether a word is used or not does.

3.2.3. Tf-idf matrix

As a further augmentation of the document term matrix, we introduce the *tf-idf form*. Tf-idf is a is a numerical statistic that is intended to reflect how important a word is to a review in the data set. It can be used to identify words in a collection of documents that are useful for determining topics in documents[13]. The principal idea is that given a term in the document set, the significance of that term is expressed as a product of the probability that it occurs within the document and the amount of information that the use of term gives on the document. How this is done, is explained in the rest of this section. We will introduce the definition of this metric and introduce some concepts from the domain of *information theory* to explain what motivates its use.

Tf-idf is an abbreviation for *term frequency - inverse document frequency*, and is obtained by multiplying the *term frequency* with the *inverse document frequency*. The term frequency is the number of times a word occurs in a document, divided by the length of the document, which is the same value used as in the mean form document term matrix. Let the term frequency in a document i for term x_v be denoted as f_{i,x_v}. The inverse document frequency is defined as

$$\text{idf}(x_v) = \log \frac{n}{n_{x_v}}, \tag{3.1}$$

where n is the number of documents in the dataset, and n_{x_v} is the number of of documents containing the term x_v. Thus we see that the inverse document frequency value for a word is constant within the data set. Taking the log of the fraction seems a bit odd. The reason for this will become clear when we introduce some concepts from information theory. Finally, the tf-idf is then obtained by multiplying the term frequency with the inverse document frequency, i.e.

$$\text{tf-idf}\left(x_v, i\right) = f_{i,x_v} \times \log \frac{n}{n_{x_v}} = \text{tf} \times \text{idf} \tag{3.2}$$

The tf-idf from matrix will then look like this:

binary_indicator	blunt	close	garbag	good	im	product	shave	...
0	0.56	0.13	0.49	0.13	0.53	0.20	0.05	...
1	0	0	0	0	0	0	0.10	...
0	0	0	0.31	0	0	0	0	...
0	0	0	0	0	0	0	0.07	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Figure 3.5: Example of a possible tf-idf form matrix. Numbers are rounded down to two decimals.

What we see is that high values are allocated when a term is used often within the document, and not that often in other documents. The tf-idf value increases proportionally to the the number of times a term is used within the review and is adjusted for how frequently a word appears in the data set anyway.

Now, the motivation for this choice of metric. We will need to introduce a few concepts from the domain of information theory. Let x_i and y_j be two distinct events from finite event spaces X and Y. Assume a joint probability distribution $P\left(x_i, y_j\right)$ is given for $x_i \in X$ and $y_j \in Y$. By the definition of the marginal distribution it is true that

$$P\left(x_i\right) = \sum_{y_j \in Y} P\left(x_i, y_j\right) \tag{3.3}$$

$$P\left(y_j\right) = \sum_{x_i \in X} P\left(x_i, y_j\right) \tag{3.4}$$

In information theory, a basic entity is the *amount of information* given by an observation.

Definition 3.1 (Amount of information)
The amount of information given by an event x_i from a finite event space X is given by the log of the inverse of the probability, i.e.

$$\log\left(1/P\left(x_i\right)\right) = -\log P\left(x_i\right) \tag{3.5}$$

Generally, we work with random variables in the event spaces. The amount of information expected for these random variables is called the *self-entropy*.

Definition 3.2 (Self-entropy)
Let $\mathscr{X}$ be a random variable representing distinct events in X. The self entropy $\mathscr{H}\left(\mathscr{X}\right)$ is then given by

$$\mathscr{H}\left(\mathscr{X}\right) := -\sum_{x_i \in X} P\left(x_i\right) \log P\left(x_i\right) \tag{3.6}$$

Similarly, we have for a random variable $\mathscr{Y}$ representing distinct events in Y that

$$\mathscr{H}\left(\mathscr{Y}\right) = -\sum_{y_j \in Y} P\left(y_j\right) \log P\left(y_j\right) \tag{3.7}$$

The self-entropy expresses the degree of uncertainty about which event will occur in a future observation. Note that the self-entropy naturally increases for larger numbers of events with equally likely probabilities.

Definition 3.3 (Pairwise mutual information)
Let x_i and y_j be two events from probability spaces X and Y. The pairwise mutual information $\mathscr{M}\left(x_i, y_j\right)$ is then given by

$$\mathscr{M}\left(x_i, y_j\right) = \log \frac{P\left(x_i, y_j\right)}{P\left(x_i\right) P\left(y_j\right)} \tag{3.8}$$

We see that pairwise mutual information between x_i and y_j is the difference between the amounts of information based on the joint probability $P(x_i, y_j)$ and the expected probability when the independence of the two events are assumed. For random variables $\mathscr{X}$ and $\mathscr{Y}$, we introduce the *expected mutual information*.

Definition 3.4 (Expected mutual information)
Let $\mathscr{X}$ and $\mathscr{Y}$ be random variables representing distinct events in X and Y. The expected mutual information between $\mathscr{X}$ and $\mathscr{Y}$, denoted as $\mathscr{J}\left(\mathscr{X};\mathscr{Y}\right)$, is then given by

$$\mathscr{J}\left(\mathscr{X};\mathscr{Y}\right) = \sum_{x_i \in X} \sum_{y_j \in Y} P\left(x_i, y_j\right) \mathscr{M}\left(x_i, y_j\right) \tag{3.9}$$

The expected mutual information represents the uncertainty about either $\mathscr{X}$ or $\mathscr{Y}$ when the other is known. We can express the expected mutual information in terms of self-entropy. First, we introduce the concept of conditional entropy.

Definition 3.5 (Conditional entropy)
Let $\mathscr{X}$ and $\mathscr{Y}$ be random variables representing distinct events in X and Y. The conditional entropy $\mathscr{H}\left(\mathscr{X}|\mathscr{Y}\right)$ is then given by

$$\mathscr{H}\left(\mathscr{X}|\mathscr{Y}\right) := -\sum_{y_j \in Y} \sum_{x_i \in X} P\left(x_i, y_j\right) \log \frac{P\left(x_i, y_j\right)}{P\left(y_j\right)} \tag{3.10}$$

Now we show how the we can express the expected mutual information in terms of entropy. This derivation will also motivate the choice for the definition of conditional entropy as in Definition 3.5.

$$\mathscr{J}\left(\mathscr{X};\mathscr{Y}\right) = \sum_{x_i \in X} \sum_{y_j \in Y} P\left(x_i, y_j\right) \mathscr{M}\left(x_i, y_j\right) \tag{3.11}$$

$$= \sum_{x_i \in X} \sum_{y_j \in Y} P\left(x_i, y_j\right) \log \frac{P\left(x_i, y_j\right)}{P\left(x_i\right) P\left(y_j\right)} \tag{3.12}$$

$$= \sum_{x_i \in X} \sum_{y_j \in Y} P\left(x_i, y_j\right) \log \frac{P\left(x_i | y_j\right) P\left(y_j\right)}{P\left(x_i\right) P\left(y_j\right)} \tag{3.13}$$

$$= \sum_{x_i \in X} \sum_{y_j \in Y} P\left(x_i, y_j\right) \left(\log P\left(x_i | y_j\right) - \log P\left(x_i\right)\right) \tag{3.14}$$

$$= - \sum_{x_i \in X} \sum_{y_j \in Y} P\left(x_i, y_j\right) \log P\left(x_i\right) + \sum_{x_i \in X} \sum_{y_j \in Y} P\left(x_i, y_j\right) \log P\left(x_i | y_j\right) \tag{3.15}$$

$$= - \sum_{x_i \in X} P\left(x_i\right) \log P\left(x_i\right) + \sum_{y_j \in Y} P\left(y_j\right) \sum_{x_i \in X} P\left(x_i | y_j\right) \log P\left(x_i | y_j\right) \tag{3.16}$$

$$= \mathscr{H}\left(\mathscr{X}\right) - \sum_{y_j \in Y} P\left(y_j\right) \mathscr{H}\left(\mathscr{X} | y_j\right) \tag{3.17}$$

$$= \mathscr{H}\left(\mathscr{X}\right) - \mathscr{H}\left(\mathscr{X} | \mathscr{Y}\right) \tag{3.18}$$

And thus we see that for the conditional entropy it is true that:

$$\mathscr{H}\left(\mathscr{X} | \mathscr{Y}\right) := - \sum_{y_j \in Y} \sum_{x_i \in X} P\left(x_i, y_j\right) \log \frac{P\left(x_i, y_j\right)}{P\left(y_j\right)} \tag{3.19}$$

$$= - \sum_{y_j \in Y} \sum_{x_i \in X} P\left(x_i | y_j\right) P\left(y_j\right) \log \frac{P\left(x_i | y_j\right) P\left(y_j\right)}{P\left(y_j\right)} \tag{3.20}$$

$$= - \sum_{y_j \in Y} P\left(y_j\right) \sum_{x_i \in X} P\left(x_i | y_j\right) \log P\left(x_i | y_j\right) \tag{3.21}$$

$$= \sum_{y_j \in Y} P\left(y_j\right) \mathscr{H}\left(\mathscr{X} | y_j\right) \tag{3.22}$$

where $\mathscr{H}\left(\mathscr{X} | y_j\right)$ is simply defined as to how we defined self-entropy in Definition 3.2, and where we used that

$$\mathscr{H}\left(\mathscr{X} | y_j\right) = - \sum_{x_i \in X} P\left(x_i | y_j\right) \log P\left(x_i | y_j\right) = - \sum_{x_i \in X} P\left(x_i | y_j\right) \log \frac{P\left(x_i, y_j\right)}{P\left(y_j\right)} \tag{3.23}$$

Now we have established enough knowledge from the domain of information theory. We take a step back to the world of documents and terms. We have a set of documents $\{1, \ldots, n\}$ and a set of distinct terms in these documents $\{x_1, \ldots, x_V\}$. We will look at the event of selecting a random document i and the event of selecting a random term x_v in the data set. These two influence each other. 'Niet onafhankelijk, maar wel als je kijkt naar het voorkomen van een woord in zekere documenten/het voorkomen van documenten met een zeker woord.'

Let $\mathscr{N}$ be the random variable over the events of selecting a document from $\{1, \ldots, n\}$ and let $\mathscr{X}$ be the random variable over the events of selecting a terms from $\{x_1, \ldots, x_V\}$. Consider the event of looking at a random term x_v. This term only appears in a certain number of the documents

When selecting a document in the data set at random, clearly $P(i) = \frac{1}{n}$, for a document $i \in \{1, \ldots, n\}$. Note that n represents the total number of documents in the data set here, not the nth document. Then we have

that for each document, the amount of information calculated is identically given by $-\log\left(\frac{1}{n}\right)$. Thus, for the self-entropy of $\mathscr{N}$ we have

$$\mathscr{H}\left(\mathscr{N}\right) = -\sum_{i\in\{1,\ldots,n\}} P\left(i\right)\log P\left(i\right) = -n\frac{1}{n}\log\frac{1}{n} = -\log\frac{1}{n} \tag{3.24}$$

Now, consider the situation where we are only interested in the subset of documents that contain x_v, $v \in \{1,\ldots,V\}$. Let the number of documents in this subset be n_v. Then, when selecting a document in this subset of the data at random, we have for the entropy of $\mathscr{N}$ given x_v that

$$\mathscr{H}\left(\mathscr{N}|x_v\right) = -\sum_{i\in\{1,\ldots,n:x_v\in i\}} P\left(i|x_v\right)\log P\left(i|x_v\right) = -n_v\frac{1}{n_v}\log\frac{1}{n_v} = -\log\frac{1}{n_v} \tag{3.25}$$

Now, look at the event of randomly selecting a term x_v from the whole data set. When we have a document i that contains term x_v, we let $\mathrm{tf}\left(i,x_v\right)$ denote the number of times the term x_v occurs in document i. Note that this is different from the how we introduced tf earlier in this section. Let L denote the sum of the lengths of all documents $\{1,\ldots,n\}$. The probability that a specific x_v is selected is then $\sum_{i\in\{1,\ldots,n\}}\frac{\mathrm{tf}\left(i,x_v\right)}{L}$. Using this, we then have for the expected mutual information that

$$\mathscr{I}\left(\mathscr{N},\mathscr{X}\right) = \mathscr{H}\left(\mathscr{N}\right) - \mathscr{H}\left(\mathscr{N}|\mathscr{X}\right) = \sum_{x_v\in\{x_1,\ldots,x_V\}} P\left(x_v\right)\left(\mathscr{H}\left(\mathscr{N}\right) - \mathscr{H}\left(\mathscr{N}|x_v\right)\right) \tag{3.26}$$

$$= \sum_{x_v\in\{x_1,\ldots,x_V\}}\sum_{i\in\{1,\ldots,n\}}\frac{\mathrm{tf}\left(i,x_v\right)}{L}\left(-\log\frac{1}{n}+\log\frac{1}{n_v}\right) \tag{3.27}$$

$$= \frac{1}{L}\sum_{x_v\in\{x_1,\ldots,x_V\}}\sum_{i\in\{1,\ldots,n\}}\mathrm{tf}\left(i,x_v\right)\log\frac{n}{n_v} \tag{3.28}$$

Recognize that we are now taking the sum over the tf-idf values. This expression shows that summing the tf-idf of all possible terms and documents recovers the mutual information between documents and term taking into account their joint distributions. Each tf-idf value thus conveys a little bit of information attached to the combination of document and term. Furthermore, the expression can be interpreted as showing the decrease of uncertainty about the relevant documents as a result of a submitted term. Note that in this expression $\mathrm{tf}\left(i,x_v\right)$ is only a count of the word x_v in document i and this count is not divided by the total number of words in document i. When we apply tf-idf values in the rest of this thesis, we also divide by the total number of words in the document to obtain the term frequency. The conclusion of the write up above still holds.

Arguments can be made both in favour and against using the tf-idf values. It as a way to incorporate information on the relevance of the use of a term within a document set in the variables used. On the other hand, there are also reasons to think that a document classifier would perform worse using the tf-idf values. When doing classification, we will aim to have a model that recognizes which variables are common among a set of documents. Tf-idf lets variables for the common words shrink and grants higher values to the variables for those words that make the documents unique. Regardless, the empirical results in Chapter 8 will show us how the use of tf-idf in our model performs.

Now move back to the tf-idf matrix form. Aside from the regular tf-idf form of the document-term matrix, we also introduce the normalized tf-idf form. It is found by taking the values found in the tf-idf norm, and dividing them by the Euclidian norm. I.e., when we let $x_{1,\text{tf-idf}},\ldots,x_{V,\text{tf-idf}}$ be the variables as found for the tf-idf form, the variables for the normalized tf-idf form, $x_{1,\text{norm-tf-idf}},\ldots,x_{V,\text{norm-tf-idf}}$ become

$$x_{i,\text{norm-tf-idf}} = \frac{x_{i,\text{tf-idf}}}{\sum_{j=1}^{V} x_{j,\text{tf-idf}}^2}, \forall\, i\in\{1,\ldots,V\}$$

The resulting matrix becomes one of the form

binary_indicator	blunt	close	garbag	good	im	product	shave	...
0	0.46	0.1	0.40	0.11	0.44	0.17	0.04	...
1	0	0	0	0	0	0	0.03	...
0	0	0	0.20	0	0	0	0	...
0	0	0	0	0	0	0	0.10	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Figure 3.6: Example of a possible normalized tf-idf form matrix. Numbers are rounded down to two decimals.

To sum this section up, we have the following options for the forms of our variables $x_1, \ldots, x_V$:

1. Document-term matrix form.
2. Mean matrix form.
3. Indicator matrix form.
4. Tf-idf matrix form.
5. Normalized tf-idf matrix form.

Figure 3.7: The possible matrix forms for variables $x_1, \ldots, x_V$

3.3. Word count threshold

When we have obtained the variables in one of the forms as described above, we will still have another choice to make. When running our models, we will find that we have a rather large number of predictive variables. To give an indication, after preprocessing of the data and forming the variables (regardless of which form) the n775-data set gives over 2900 variables and the n1976-data set even gives more than 4000 variables. We have not yet formally introduced the models that we will run in our analysis but, although we have selected them such that they are rather well equipped to deal with these numbers of variables, these numbers of variables will still come with a high computational cost.

To counter this problem, we define the possibility of setting what we call a *word count threshold*. For each variable $x_v, v \in \{1, \ldots, V\}$, we check whether it occurs in the data set more often than this threshold. If it does not, we remove the variable. Reviews that did use the word x_v are kept in the data set of course, they just lose the value they had for the variable. Note that one review using the word x_v twice, also counts twice to reaching the word count threshold. In this thesis, we will experiment with word count thresholds of 1, 6, 11, 16 and 21. In the first case no words are thrown away, in the other words that are used respectively 5 or less times in the data set, 10 or less, 15 or less and 20 or less times.

4

Evaluation Methods

In the chapter 2 the probabilities $\widehat{p}_{i,j}$ and predictions $\widehat{d'}_{i,j}$ were defined. Now, when we have a model that can generate these probabilities and predictions, how will we we go about evaluating these results? In this chapter, we will dive into this question. First, we will briefly discuss how we evaluate using a part of our data set. Then, some basic evaluation methods will be discussed, and we will see why these do not always suit the purpose of this thesis. Lastly, we will discuss some more advanced evaluation methods, and discuss why these better suit the purpose of this thesis.

4.1. Evaluation data

Having predictions implies that we have new data, or in the context of this thesis new documents, coming in and we have a model trained on our current data set ready to launch on this new data. However, this is not the case in this thesis. There is data available and we can train models on it, but we do not actively have new data coming in. So, how can we compute predictions, when there is no new data to predict?

The solution is simple. We take some of our training data and pretend this is the new data coming in. We name this data I_{new}. A model is trained using the remainder of the data. We can then use this model to compute predictions for the documents in I_{new}. Because we also have the actual driver scores for the set I_{new}, we can use these to measure the performance of our model. This brings us to the process of *cross validation*.

4.1.1. Estimating performance using our data

Cross validation is a commonly used technique in statistical analyses. When doing a cross validation, a data set is split into complementary subsets. One subset is removed is removed from the data set and the model is trained on the part of the data that is left behind. The latter part of the data is named the *training set*, the removed subset is named the *test set*. The model is then used to compute predictions for the reviews in the test set, and the performance of our model is measured. This process is then repeated for every split of training and test set. The average of the measured performance in each split, becomes our estimated of the model performance. If n is the number of unique splits made, then we say that an *n-fold cross validation* was performed. The following picture illustrates the cross validation.

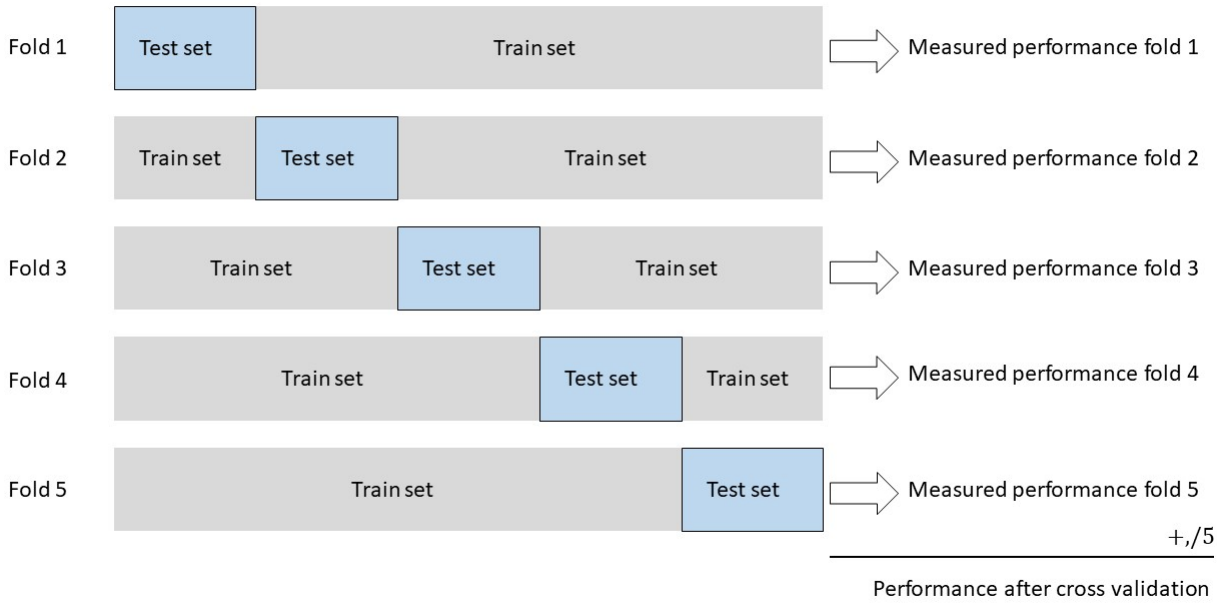


Figure 4.1: Illustration of a 5-fold cross validation.

Mathematically speaking, we name the test set I_{test} and we name the reviews left in the dataset I_{train}. A model is trained on I_{train}, and evaluated using its performance on I_{test}. . This is then repeated an k number of times, k being the number of unique subsets the data set was split in, giving k evaluation results. The average over these results is taken giving us our estimated model performance. How this evaluation is done is discussed in the remainder of this chapter.

4.2. Evaluation functions

4.2.1. A logical choice

When thinking of how to evaluate the predictions, one of the first things to come to mind is to simply count the number of correct predictions, using the ground truths, or equivalently, counting the number of wrong predictions. We introduce the misclassification error.

Definition 4.1 (Misclassification error)
Let $\widehat{d} = \left[\widehat{d}_1, \ldots, \widehat{d}_n\right]$ be a vector of n predictions for $d = \left[d_1, \ldots, d_n\right]$, where d is known. Then the misclassification error E is defined as

$$E\left(d, \widehat{d}\right) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{d_i \neq \widehat{d}_i}$$

In the context of this thesis, we assume we have n_{new} predictions, say $\widehat{d}'_{I_{\text{new}},j}$ for the values $d'_{I_{\text{new}},j}$. Then, these will look something like this:

$$\underset{d'_{I_{\text{new}},j}}{\begin{bmatrix}1\\0\\0\\0\\1\\0\\\vdots\end{bmatrix}} \qquad \underset{\widehat{d}'_{I_{\text{new}},j}}{\begin{bmatrix}1\\0\\0\\0\\1\\1\\\vdots\end{bmatrix}}$$

Figure 4.2: Example of data and predictions in the binary setting, for a driver j and documents $i \in I_{\text{new}}$.

And we can compute the misclassification error E by the formula:

$$E\left(d'_{I_{\text{new}},j}, \widehat{d}'_{I_{\text{new}},j}\right) = \frac{1}{n_{\text{new}}} \sum_{i \in I_{\text{new}}} \mathbb{1}_{d'_{i,j} \neq \widehat{d}'_{i,j}} \tag{4.1}$$

However, we will not be using the misclassification error extensively in this thesis. Why this? In Chapter 2 we saw the distribution of the driver scores in Figures 2.3 and 2.4. It is clear that when looking at one driver j, many of the reviews will be annotated as 'NA' for this particular driver. In the binary setting, we generally choose our sentiment such that $d'_{i,j} = 0$ when $d_{i,j} = NA$. This means that there is an imbalance in the (coarsened) driver scores in our data set, as $d'_{i,j}$ will equal 0 for many documents and 1 for only a few. The same holds for the documents in I_{new}, as long as they are not distributed in an extremely different manner than in the rest of the data set. Say that we have 90% of the documents in I_{new} have a 0 for the coarsened driver score. A model only giving 0s, regardless of what you ask it to predict, will then have a misclassification error of 0.1 which seems rather good, but we are of course not looking for such a model. We will want a model that is able to, at least to a certain extent, distinguish the 1s from the 0s. This means that we can even prefer a model with a higher misclassification error, when it tries to predict 1s.

Note that the misclassification error can also be applied as a measure in the multinomial setting. Unfortunately, the associated problem with the measure does as well. When we work with classes P, N and NA, the NA-class will dominate the coarsened driver scores, and relatively few drivers will be of the P and N classes.

4.2.2. Handling the class imbalance

A solution to the problem presented by the misclassification error is the so called *area under the curve* (AUC); area under the so called Receiver Operating Characteristic (ROC) curve that is. We will introduce the ROC curve and other concepts required for understanding the AUC in this section, and end with the AUC itself. The regular AUC measure is only defined for a binary classification and thus, for the rest of this subsection, we will only work in the binary setting.

Definition 4.2 (confusion matrix)
Let $\widehat{d} = \left[\widehat{d}_1, \ldots, \widehat{d}_n\right]$ *be a vector of n predictions for* $d = \left[d_1, \ldots, d_n\right]$*, where d is known and we have* $d_i \in \{0,1\}, \forall i \in \{1, \ldots, n\}$ *and* $\widehat{d}_i \in \{0,1\}, \forall i \in \{1, \ldots, n\}$*. Then*

- *True Positives (TP)* $\left(d, \widehat{d}\right) := \{i \in \{1, \ldots, n\} : d_i = \widehat{d}_i = 1\}$
- *True Negatives (TN)* $\left(d, \widehat{d}\right) := \{i \in \{1, \ldots, n\} : d_i = \widehat{d}_i = 0\}$
- *False Positives (FP)* $\left(d, \widehat{d}\right) := \{i \in \{1, \ldots, n\} : d_i = 0 \neq \widehat{d}_i = 1\}$
- *False Negatives (FN)* $\left(d, \widehat{d}\right) := \{i \in \{1, \ldots, n\} : d_i = 1 \neq \widehat{d}_i = 0\}$

These values are represented in the confusion matrix as

		d_i	
		1	0
$\widehat{d}_i$	1	TP	FP
	0	FN	TN

Figure 4.3: The confusion matrix.

Definition 4.3 (Sensitivity and specificity)
Let $\widehat{d} = \left[\widehat{d}_1, \ldots, \widehat{d}_n\right]$ *be a vector of n predictions for* $d = \left[d_1, \ldots, d_n\right]$*, where d is known and we have* $d_i \in \{0,1\}, \forall i \in \{1, \ldots, n\}$ *and* $\widehat{d}_i \in \{0,1\}, \forall i \in \{1, \ldots, n\}$*. Then,*

$$sensitivity\left(d, \widehat{d}\right) := \frac{\left|TP\left(d, \widehat{d}\right)\right|}{\left|TP\left(d, \widehat{d}\right)\right| + \left|FN\left(d, \widehat{d}\right)\right|}$$

$$specificity\left(d, \widehat{d}\right) := \frac{\left|TN\left(d, \widehat{d}\right)\right|}{\left|FP\left(d, \widehat{d}\right)\right| + \left|TN\left(d, \widehat{d}\right)\right|}$$

To place this in the context of this thesis, again assume we have a vector with driver scores $d'_{I_{\text{new}},j}$, and a vector with predicted values $\widehat{d'}_{I_{\text{new}},j}$. These will be of the following form:

$$\underset{d'_{I_{\text{new}},j}}{\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}} \quad \underset{\widehat{d}'_{I_{\text{new}},j}}{\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ \vdots \end{bmatrix}}$$

Figure 4.4: Example of data and predictions in the binary setting, for a driver j and documents $i \in I_{\text{new}}$.

Having these values, we can compute the TP, TN, FP and FN, and the specificity and sensitivity. Sensitivity is also know as *True Positive Rate* and specificity as *True Negative Rate*. Why are these values important to us? We will explain. When, in the context of this thesis, a model is created, it gives us a vector of estimated probabilities $\widehat{p}_{I_{\text{new}},j}$, and the predictions $\widehat{d}'_{I_{\text{new}},j}$ are derived from these probabilities by setting a threshold value t.

Now, when looking at the definitions of specificity and sensitivity, and TP, FP, FN and TN, one can see the effect of changing the threshold value t. Lowering the threshold value will lead to more 1's in the vector $\widehat{d}'_{I_{\text{new}},j}$. This means that the values TP and/or FP will increase and FN and/or TN will decrease. Let us illustrate with an example:

$$\underset{\widehat{p}_{I_{\text{new}},j}}{\begin{bmatrix}0.90\\0.80\\0.73\\0.62\\0.53\\0.31\\0.31\\0.12\end{bmatrix}} \quad \underset{\widehat{d}'_{I_{\text{new}},j}}{\begin{bmatrix}1\\1\\1\\1\\1\\0\\0\\0\end{bmatrix}} \quad \underset{d'_{I_{\text{new}},j}}{\begin{bmatrix}1\\1\\1\\0\\1\\0\\1\\0\end{bmatrix}}$$

Figure 4.5: Example of data set, computed probabilities and predictions in the binary setting, for a driver j and documents $i \in I_{\text{new}}$, with $n_{\text{new}} = 8$, with a set threshold value of $t = 0.5$.

In this case, we find a sensitivity of 0.8 and a specificity of 0.67. Note in this case, it does not matter whether we choose the threshold as $t = 0.52$ or $t = 0.47$ or any value in the range $[0.32, 0.52]$; the values for sensitivity and specificity stay the same when using this choice of I_{new}. As long as the threshold value stays between the same two probabilities, there is no change. Now, we decrease the threshold to $t = 0.3$ and get the following situation:

$$\underset{\widehat{p}_{I_{\text{new}},j}}{\begin{bmatrix}0.90\\0.80\\0.73\\0.62\\0.53\\0.31\\0.31\\0.12\end{bmatrix}} \quad \underset{\widehat{d}'_{I_{\text{new}},j}}{\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\0\end{bmatrix}} \quad \underset{d'_{I_{\text{new}},j}}{\begin{bmatrix}1\\1\\1\\0\\1\\0\\1\\0\end{bmatrix}}$$

Figure 4.6: Example of data set, computed probabilities and predictions in the binary setting, for a driver j and documents $i \in I_{\text{new}}$, with $n_{\text{new}} = 8$, with a set threshold value of $t = 0.3$.

In this case, we find a sensitivity of 1 and a specificity of 0.33. We see that as we decrease the threshold value t, there is a monotonic increase in sensitivity and a monotonic decrease in specificity. The opposite holds as well; when we increase the threshold value t, there is a monotonic decrease in sensitivity and a monotonic increase in specificity. Thus a (monotonic) increase in either sensitivity or specificity leads to a (monotonic) decrease in the other. Thus, we see that when we let $\widehat{d}$ be a known function of $\widehat{p}_{I_{\text{new}},j}$ and t such that

$$\widehat{d}\left(\widehat{p}_{I_{\text{new}},j}, t\right) = \widehat{d}'_{I_{\text{new}},j} \tag{4.2}$$

then

$$\text{sensitivity}\left(d'_{I_{\text{new}},j}, \widehat{d}'_{I_{\text{new}},j}\right) = \text{sensitivity}\left(d'_{I_{\text{new}},j}, \widehat{d}\left(\widehat{p}_{I_{\text{new}},j}, t\right)\right) \tag{4.3}$$

$$\text{specificity}\left(d'_{I_{\text{new}},j}, \widehat{d}'_{I_{\text{new}},j}\right) = \text{sensitivity}\left(d'_{I_{\text{new}},j}, \widehat{d}\left(\widehat{p}_{I_{\text{new}},j}, t\right)\right) \tag{4.4}$$

Or, we will also simply write

$$\text{sensitivity}\left(d'_{I_{\text{new}},j}, \widehat{p}_{I_{\text{new}},j}, t\right) \tag{4.5}$$

$$\text{specificity}\left(d'_{I_{\text{new}},j}, \widehat{p}_{I_{\text{new}},j}, t\right) \tag{4.6}$$

where the use of the function of $\widehat{d}$ as function of the last two arguments is implied. Thus, we see that the sensitivity and specificity are, given a set of documents, functions of the coarsened driver scores and estimated probabilities for the documents, and the threshold value t. This and the earlier mentioned relationship between the sensitivity and specificity are characterised through the *ROC curve*.

4.2.3. The ROC curve

Definition 4.4 (ROC curve)
Let $d = \left[d_1, \ldots, d_n\right]$ *be a known vector of length* n *where* $d_i \in \{0,1\}, \forall\, i \in \{1, \ldots, n\}$. *Let* $p = \left[p_1, \ldots, p_n\right]$ *be a vector of probabilities, such that* $0 \leq p_i \leq 1$. *Let* t, *the threshold value, be a variable in* $[0,1]$.

The Receiver Operating Characteristic (ROC) curve[1] *plots sensitivity against* $1-$*specificity. It is created by computing the sensitivity*(d, p, t) *and* $1-$*specificity*(d, p, t) *for all values of* $t \in [0,1]$, *and then connecting the points found by plotting these two against each other.*

An example of the ROC curve is:

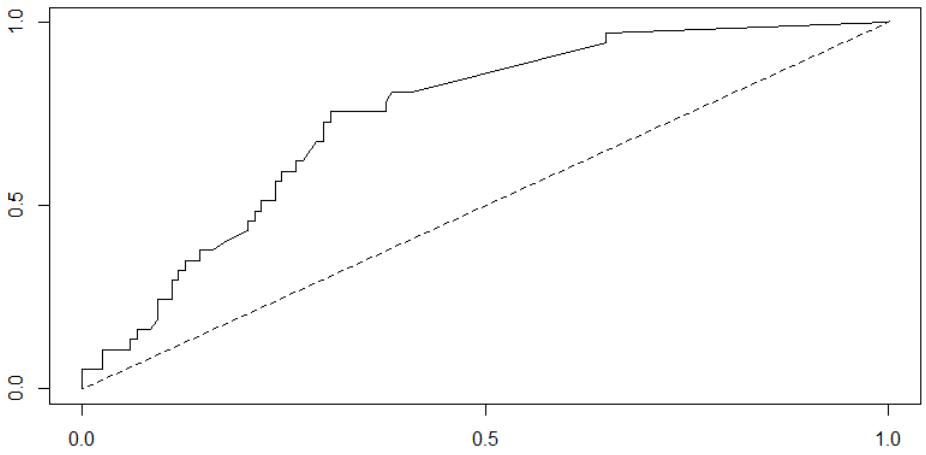


Figure 4.7: Example of an ROC curve. On the y-axis is depicted sensitivity, bounded between 0 and 1. On the x-axis is depicted 1−specificity, bounded between 0 and 1. The dashed line depicts models that score as good as a random guess.

Note that we have sensitivity on the y-axis and $1-$specificity on the x-axis. This is the standard for ROC curves. The reason for this may not be clear immediately. Note that

$$\text{sensitivity} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}|}, \tag{4.7}$$

which shows why sensitivity is also known as the *true positive rate*. We also have

$$1 - \text{specificity} = 1 - \frac{|\text{TN}|}{|\text{FP}| + |\text{TN}|} = \frac{|\text{FP}|}{|\text{FP}| + |\text{TN}|}, \tag{4.8}$$

which shows that $1-$specificity is the *false positive rate*. Thus, the ROC curve is essentially plotting *the probability of predicting a real positive as a positive* versus *the probability of predicting a real negative as a positive*, which is also the reason for this being the standard in literature.

In the context of this thesis, the ROC curve is computed using a set of new documents I_{new}, the probabilities computed for this set, $\widehat{p}_{I_{\text{new}},j}$, and the corresponding real values, $d'_{I_{\text{new}},j}$. We want the curve to be as close to the top left as possible, where sensitivity and specificity both equal 1. The ROC curve actually reaching this point means that we can set a threshold value, within the vector of probabilities $\widehat{p}_{I_{\text{new}},j}$ that gives perfect distinction between the corresponding true 1's and true 0's. This will be very hard or impossible to achieve when creating models with actual data, but we will try to get as close as possible.

When a model is created that simply guesses, this model will end up as a point somewhere on the straight line running from the bottom left to the upper right in Figure 4.7. All models that simply guess are on this line, starting with models always predicting 0 in the bottom left, going to models predicting more and more 1s and eventually only predicting 1s when being in the top right. It is not impossible for a model to be beneath this line, but that would mean the model is not performing. It could be that the data on which the model

[1] The ROC curve was first developed during World War 2, within the context of determining if a blip on a radar screen represented a ship or an extraneous noise. *Radar receiver operators* used the ROC curve to set the threshold for military action.

was trained is very different from the data on which the model was validated, and the found specificity and sensitivity are, for any threshold, worse than those that would be obtained by a random guess. Of course, it is always a possibility that the model is beneath the straight line, because it is simply rubbish. In fact, a simple improvement of a model beneath the straight line would be to always predict the opposite of what the model predicts, which would give a model running above the straight line.

Remember that earlier we had the misclassification error, which is a relatively simple statistic. We can take the predictions of a model and the corresponding the true values, compute the misclassification error and then compare it with the same results for a different model. Now, in the context of the ROC curve, it is clear that we want to get as close as possible to both a sensitivity and a specificity of 1, but we will still want a way of expressing this desire in one value as a way to compare models. One way of doing is, is by computing the *area under the curve*, abbreviated as *AUC*.

4.2.4. Summarizing the result in one value

The AUC is defined as the area under the ROC curve, hence the name. This means that the AUC gives us information about the performance of a model before setting a threshold t. This, because every point on the ROC curve is given by a combination of specificity and sensitivity at a certain value of t. When computing the area under this curve, which is formed for given vectors d and p, you are thus essentially integrating over t. Before defining the AUC, we need to go back to the concepts of sensitivity and specificity first.

Let $d = \left[d_1, \ldots, d_n\right]$ be a known vector of length n where $d_i \in \{0,1\}, \forall i \in \{1,\ldots,n\}$. Let p be a vector of n probabilities. In Definition 4.3 the sensitivity and specificity were introduced as

$$\text{sensitivity} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}|} \tag{4.9}$$

$$\text{specificity} = \frac{|\text{TN}|}{|\text{FP}| + |\text{TN}|} \tag{4.10}$$

with TP, FN, TN and FP functions of d and a vector $\widehat{d}$ of predictions. Within this proof, we do not have $\widehat{d}$ fixed yet, as we have p and are free to choose threshold value t in $[0,1]$. Note that, regardless of how we choose t, we have that

$$|\text{TP}| + |\text{FN}| = \left|\left\{i \in \{1,\ldots,n\} : d_i = 1\right\}\right| \tag{4.11}$$

$$|\text{FP}| + |\text{TN}| = \left|\left\{i \in \{1,\ldots,n\} : d_i = 0\right\}\right| \tag{4.12}$$

And for a given t, we have that

$$\text{TP} = \left\{i \in \{1,\ldots,n\} : d_i = 1 \wedge p_i^* \geq t\right\} \tag{4.13}$$

$$\text{FP} = \left\{i \in \{1,\ldots,n\} : d_i = 0 \wedge p_i^* \geq t\right\} \tag{4.14}$$

, where p_i^* is the ith entry of p. Thus, we see that, given vectors d and p, the True Positive Rate (=sensitivity) and False Positive Rate (=1−specificity) are functions of t, as

$$\text{TPR}(t) = \frac{\left|\left\{i \in \{1,\ldots,n\} : d_i = 1 \wedge p_i^* \geq t\right\}\right|}{\left|\left\{i \in \{1,\ldots,n\} : d_i = 1\right\}\right|} \tag{4.15}$$

$$\text{FPR}(t) = \frac{\left|\left\{i \in \{1,\ldots,n\} : d_i = 0 \wedge p_i^* \geq t\right\}\right|}{\left|\left\{i \in \{1,\ldots,n\} : d_i = 0\right\}\right|} \tag{4.16}$$

Definition 4.5 (Area under the curve)
Let $d = \left[d_1, \ldots, d_n\right]$ be a known vector of length n where $d_i \in \{0,1\}, \forall\, i \in \{1,\ldots,n\}$. Let p be a vector of n probabilities. Let the True Positive Rate (TPR), based on d and p, be a function of t. The AUC is defined as

$$AUC\,(d,p) := \int_0^1 TPR\,(t)\,dt \tag{4.17}$$

Thus, for our goal of evaluating our results, the use of the AUC relieves us from the obligation of setting a threshold t, and only $d'_{I_{\text{new}},j}$ and $\widehat{p}_{I_{\text{new}},j}$ are needed for its computation. When ultimately computing $\widehat{d'}_{I_{\text{new}},j}$, t will still have to be chosen.

In general, a model with a high AUC value is desirable, and the value is bounded between 0 and 1. When only considering models better than a random guess, it is bounded between 0.5 and 1. Now, since the AUC equals the area under the AUC curve, it can be calculated by piecewise integration of this curve. However, there is another method; it can be shown that the AUC equals the probability of scoring a true 1 higher than a true 0. In the context of this thesis, scoring means the probability allocated to a document. We will illustrate what this means.

Again, let p a vector of length n of probabilities, i.e. every entry is bounded between 0 and 1. So far, we have always seen its entries simply as numbers. Now, we will deviate from this concept, and assume that its entries are outcomes of random variables $p_i, i \in \{1,\ldots,n\}$. Let $d = \left[d_1, \ldots, d_n\right]$ be a known vector of length n where $d_i \in \{0,1\}$. We then say that p_i is a random variable whose probability density function is f_1 when $d_i = 1$ and f_0 when $d_i = 0$. It then holds that

$$P\left(p_i \le x | d_i = 1\right) = \int_{-\infty}^{x} f_1(u)\,\mathrm{d}u \tag{4.18}$$

$$P\left(p_i \le x | d_i = 0\right) = \int_{-\infty}^{x} f_0(u)\,\mathrm{d}u \tag{4.19}$$

Let X_1 and X_0 be independent draws from f_1 and f_0 respectively. It then holds that

$$P(X_1 > X_0) = \int_{-\infty}^{\infty}\int_{x_0}^{\infty} f_0(x_0) f_1(x_1)\,\mathrm{d}x_1\,\mathrm{d}x_0 \tag{4.20}$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbb{1}_{\{\tau \ge t\}} f_1(\tau) f_0(t)\,\mathrm{d}\tau\,\mathrm{d}t \tag{4.21}$$

Keep in mind that X_1 is a draw from p_i when $d_i = 1$ and X_0 is a draw from p_i when $d_0 = 1$. We therefore say that $P(X_1 > X_0)$ is the probability of scoring a true 1 greater than a true 0. In the context of this thesis, the density functions f_1 and f_0 are not given. We will try to approach these densities using the (finite) samples d and p of size n, for which we compute the AUC. Then, we can show that the AUC approaches this probability $P(X_1 > X_0)$.

Theorem 4.1 (AUC)
Let $d = \left[d_1, \ldots, d_n\right]$ be a known vector of length n where $d_i \in \{0,1\}, \forall\, i \in \{1,\ldots,n\}$. Let p be a vector of n outcomes of independent random variables p_i whose probability density function is f_1 when $d_i = 1$ and f_0 when $d_i = 0$. Let X_1 and X_0 be independent draws from f_1 and f_0 respectively. Let the AUC be the area under the ROC curve formed by d and p, as in Definition 4.4. Then

$$AUC\,(d,p) \approx P(X_1 > X_0)$$

We will prove this result in the remainder of this section. We have that

$$P\left(p_i \le x | d_i = 1\right) = \int_{-\infty}^{x} f_1(u)\,\mathrm{d}u \tag{4.22}$$

$$P\left(p_i \le x | d_i = 0\right) = \int_{-\infty}^{x} f_0(u)\,\mathrm{d}u \tag{4.23}$$

Given vectors d and p, the TPR and FPR are functions of t, as

$$\text{TPR}(t) = \frac{\left|\left\{i \in \{1,\ldots,n\} : d_i = 1 \wedge p_i^* \geq t\right\}\right|}{\left|\left\{i \in \{1,\ldots,n\} : d_i = 1\right\}\right|} \tag{4.24}$$

$$\text{FPR}(t) = \frac{\left|\left\{i \in \{1,\ldots,n\} : d_i = 0 \wedge p_i^* \geq t\right\}\right|}{\left|\left\{i \in \{1,\ldots,n\} : d_i = 0\right\}\right|} \tag{4.25}$$

In the following, we will need the strong law of large numbers.

Theorem 4.2 (Strong law of large numbers)
Let $X_1, X_2, \ldots$ be an infinitely long sequence of outcomes of integrable random variables that all have expected value μ and let $\overline{X}_n = \frac{1}{n}\left(X_1 + X_2 + \ldots\right)$. Then

$$\overline{X}_n \xrightarrow{a.s.} \mu \tag{4.26}$$

In the setting of our proof we consider

$$I_{i,1}(t) = \mathbb{1}_{p_i^* \geq t}, \qquad i \in \{i \in \{1,\ldots,n\} : d_i = 1\} \tag{4.27}$$

$$I_{i,0}(t) = \mathbb{1}_{p_i^* \geq t}, \qquad i \in \{i \in \{1,\ldots,n\} : d_i = 0\} \tag{4.28}$$

as the outcomes of random variables and we let $\overline{I_{i,1}}(t)$ and $\overline{I_{i,0}}(t)$ be the sample averages (as $\overline{X}_n$ in Definition 4.2). Using Equations 4.24 and 4.25 we then have that

$$\text{TPR}(t) = \overline{I_{i,1}}(t) \tag{4.29}$$

$$\text{FPR}(t) = \overline{I_{i,0}}(t) \tag{4.30}$$

Now, we wish to apply the law of large numbers, which means letting n go to infinity. In our context, this means expanding the vectors p and d. In doing this, assume that the fractions of $d_i = 0$ and $d_i = 1$ in vector d stay the same, and entries p_i^* are draws from p_i, $i \in \{1,\ldots,n\}$. Since p_i^* is an outcome of the independent random variable p_i, we see that

$$\text{TPR}(t) \xrightarrow{\text{a.s.}} P\left(p_i \geq t | d_i = 1\right) \tag{4.31}$$

$$\text{FPR}(t) \xrightarrow{\text{a.s.}} P\left(p_i \geq t | d_i = 0\right) \tag{4.32}$$

as n goes to infinity.

If we then let $\text{FPR}(t) = x$, then $t = \left(\text{FPR}^{-1}(x)\right)$. One needs to be careful about defining the inverse function of FPR here. By Equation 4.25 FPR is a step function, meaning that different values of t can give the same FPR, i.e. the FPR is a non-injective function. We thus let the inverse be defined as $\text{FPR}^{-1}(x) = \inf\left(\{t : \text{FPR}(t) = x\}\right)$.

We then find that

$$\text{AUC}(d,p) = \int_0^1 \text{TPR}\left(\text{FPR}^{-1}(x)\right) \mathrm{d}x = \int_\infty^{-\infty} \text{TPR}(t)\text{FPR}'(t)\,\mathrm{d}t \tag{4.33}$$

$$\to -\int_{-\infty}^{\infty}\left(1 - \int_{-\infty}^{t} f_1(\tau)\,\mathrm{d}\tau\right)\cdot\left(-f_0(t)\right)\mathrm{d}t \tag{4.34}$$

$$= \int_{-\infty}^{\infty}\int_{t}^{\infty} f_1(\tau)\,\mathrm{d}\tau\, f_0(t)\,\mathrm{d}t \tag{4.35}$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbb{1}_{\{\tau \geq t\}} f_1(\tau) f_0(t)\,\mathrm{d}\tau\,\mathrm{d}t \tag{4.36}$$

Here, in the first step the transform $\text{FPR}(t) = x$ was used, such that $\text{TPR}\left(\text{FPR}^{-1}(x)\right) = \text{TPR}\ (t)$. Now, say that we take a draw X_1 from f_1 and a draw X_0 from f_0, we then have that

$$P(X_1 > X_0) = \int_{-\infty}^{\infty}\int_{x_0}^{\infty} f_0(x_0) f_1(x_1)\,\mathrm{d}x_1\,\mathrm{d}x_0 \tag{4.37}$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbb{1}_{\{\tau \geq t\}} f_1(\tau) f_0(t)\,\mathrm{d}\tau\,\mathrm{d}t \tag{4.38}$$

We see that Equation 4.36= Equation 4.38, and thus we have that

$$\text{AUC}(d,p) \to P\left(X_1 > X_0\right) \tag{4.39}$$

Given these, results, we conclude that given n sufficiently large, we have that

$$\text{AUC}(d,p) \approx P\left(X_1 > X_0\right) \tag{4.40}$$

q.e.d.

Theorem 4.1 gives us that the AUC equals the probability of scoring a true 1 greater than a true 0. In this thesis we will approach this probability, and thus by direct implication the AUC, using results achieved. First, we introduce an adapted indicator function, say $\tilde{1}$, defined as

$$\tilde{1}(x) := 1 - \mathbb{1}_{\{x<0\}} - \frac{1}{2}\mathbb{1}_{\{x=0\}} \tag{4.41}$$

So that if $x > y$ we have $\tilde{1}\left(x-y\right) = 1$, if $x < y$ then $\tilde{1}\left(x-y\right) = 0$ and if $x = y$ then $\tilde{1}\left(x-y\right) = 0.5$.

Now, in the context of this thesis, say that we have a model and there are n_{new} documents serving as our test set. Name this set I_{new}. Then, for a driver j, we will compute the estimated probability $\widehat{p}_{n_{\text{new}},j}$ to base our prediction for $d'_{n_{\text{n}},j}$ on.

Now, say that for $I_+ = \{i \in I_{\text{new}} : d'_{i,j} = 1\}$ and $I_- = \{i \in I_{\text{new}} : d'_{i,j} = 0\}$ and that $n_+ = \left|I_+\right|$ and $n_- = \left|I_-\right|$. Then we have that

$$\text{AUC} = \frac{1}{n_+ n_-}\sum_{i^+ \in I_+}\sum_{i^- \in I_-} \tilde{1}\left(\widehat{p}_{i^+,j} - \widehat{p}_{i^-,j}\right) \tag{4.42}$$

So that the AUC is given by the fraction of reviews for which $d'_{i,j} = 1$ that are scored higher than the reviews for which $d'_{i,j} = 0$. In the case that we have two reviews, i^+ and i^-, such that $d'_{i^+,j} = 1$ and $d'_{i^-,j} = 0$, and they are scored equally, i.e. $\widehat{p}_{i^+,j} = \widehat{p}_{i^-,j}$, the adapted indicator function $\tilde{1}$ returns 0.5 as a compromise between 1 (i^+ being scored greater) and 0 (i^- being scored greater). A small example with multiple results will illustrate what this means:

$$\underset{\widehat{p}_{I_{new},j}}{\begin{bmatrix} 0.90 \\ 0.80 \\ 0.73 \\ 0.62 \\ 0.53 \\ 0.31 \\ 0.31 \\ 0.12 \end{bmatrix}} \quad \underset{d'_{I_{new},j}}{\begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}}$$

Figure 4.8: Example of data set and computed probabilities in the binary setting, for a driver j and documents $i \in I_{new}$, with $n_{new} = 8$.

We have that $I_+ = \{0.90, 0.80, 0.73, 0.53\}$ and $I_- = \{0.62, 0.31, 0.31, 0.12\}$. Thus, using our result 4.42, we can compute our estimate of the AUC as

$$\text{AUC} = \frac{1}{n_+ n_-}\sum_{i^+ \in I_+}\sum_{i^- \in I_-} \tilde{1}\left(\widehat{p}_{i^+,j} - \widehat{p}_{i^-,j}\right) \tag{4.43}$$

$$= \frac{1}{16}(4+4+4+3) = 0.9375 \tag{4.44}$$

4.2.5. A multinomial AUC generalization

In the previous section we introduced the concepts of sensitivity and specificity, the ROC curve and the AUC. The AUC might be an effective measure for the binary setting, but we are still looking for a similar measure in the multinomial setting. Several articles have been written on multinomial generalizations of the AUC. The most widely used measure is called the M, introduced by Hand & Till (2001) [16]. In a recent paper by Kleiman & Page (2019)[14] it is noted that M has its shortcoming though, and a new measure named the AUC_μ has been introduced.

The AUC_μ has been developed such that the properties of the AUC deemed most critical by Kleiman and Page to its use and interpretation are preserved. These are

Property 1. If a model gives the correct label the highest probability on every example, then AUC=1.

Property 2. Random guessing on examples yields AUC=0.5.

Property 3. AUC is insensitive to class skew (meaning that the class distribution, i.e. having more instances from one class than another, does not affect the value of the AUC).

These three properties are all met by the AUC estimate as given in Equation 4.42, and we wish to see these properties also met by the AUC_μ. We will demonstrate how the AUC_μ is constructed.

In the multinomial setting the outcome $d'_{i,j}$ is a draw from the categorical distribution with parameters $p_{i,j} := \left[\left(p_{i,j}\right)_1, \ldots, \left(p_{i,j}\right)_P\right]$, where $p_{i,j}$ must be in the $(P-1)$-simplex. In this thesis, we then compute $\widehat{p}_{i,j}$ as an estimate for $p_{i,j}$. A prediction $\widehat{d'}_{i,j}$ for $d'_{i,j}$ is then formulated, based on in which part of the $(P-1)$-simplex $\widehat{p}_{i,j}$ is.

In Chapter 2 we introduced the concepts of the partition matrix and decision boundary (Definitions 2.3 and 2.4 respectively). As a quick help, given a misclassification matrix A, the decision boundary between classes q_1 and q_2 is given by the set of choices p in the$(P-1)$-simplex such that

$$A_{q_1,\cdot} \cdot p = A_{q_2,\cdot} \cdot p \tag{4.45}$$

, where $A_{q_1,\cdot}$ denotes the q_1th row of A. Having this, we can look at a way of ranking p in the multinomial setting. When using the AUC in the binary setting, p is 1-dimensional and the greater it is, the higher it is ranked. Now to us the task of extending this notion to the multinomial setting. Equation 4.45 describes the decision boundary as a set of possible choices for p for which the expected cost of labeling a class as q_1 equals the expected cost of labeling a class as q_2. Define

$$v_{q_1,q_2} := A_{q_1,\cdot} - A_{q_2,\cdot} \tag{4.46}$$

Then it holds that the equation of the hyperplane giving the decision boundary is given by $v_{q_1,q_2} \cdot p = 0$, which is an equivalent formulation of Equation 4.45. This decision boundary divides the $(P-1)$-simplex in two regions, one region where we label $\widehat{d} = q_1$ and one region where we label $\widehat{d} = q_2$. If $v_{q_1,q_2} \cdot p$ is positive, it is more costly to label $\widehat{d} = q_1$, and when $v_{q_1,q_2} \cdot p$ is negative, it is more costly to label $\widehat{d} = q_2$. Thus, we see that v_{q_1,q_2} provides a way to rank points in the simplex in terms of their cost difference between assignment of class q_1 and assignment of class q_2. Note that this means that while we can rank all points in the simplex, we can so far only do this when given the choice between two classes, q_1 and q_2.

Having said that, we look at the situation where we are given two documents, say i_1 and i_2 and have computed the estimated probabilities, respectively $\widehat{p}_{i_1,j}$ and $\widehat{p}_{i_2,j}$ for driver j. Without loss of generality, let the true classes be $d'_{i_1,j} = q_1$ and $d'_{i_2,j} = q_2$, $q_1, q_2 \in \{1, \ldots, P\}$. Let $y_{1,j}$ and $y_{2,j}$ be the vertices in the $(P-1)$-simplex for classes q_1 and q_2 respectively for driver j. Then we have that $y_{1,j} = e_{q_1}$ and $y_{2,j} = e_{q_2}$, where e_k is the vector of length P with 1 as the kth element, and 0 for all other elements. Then, take $v_{q_1,q_2} = A_{q_1,\cdot} - A_{q_2,\cdot}$ as the normal vector to the decision boundary. $v_{q_1,q_2} \cdot y_{1,j}$ and $v_{q_1,q_2} \cdot y_{2,j}$ are then the unscaled distances of our class vertices to the hyperplane.

These distances provide us the 'correct' orientation of two points projected onto v_{q_1,q_2}. Using this, we can define a correct ranking for $\widehat{p}_{i_1,j}$ and $\widehat{p}_{i_2,j}$. If $v_{q_1,q_2} \cdot y_{1,j} - v_{q_1,q_2} \cdot y_{2,j} > 0$, then we say that $\widehat{p}_{i_1,j}$ and $\widehat{p}_{i_2,j}$ are correctly ranked if $v_{q_1,q_2} \cdot \widehat{p}_{i_1,j} - v_{q_1,q_2} \cdot \widehat{p}_{i_2,j} > 0$. This concept brings us to the orientation function.

Definition 4.6 (Orientation function)
Let us have a classification problem with P classes with partition matrix A, and let two classes be $q_1, q_2 \in \{1,\dots,P\}, q_1 \neq q_2$. Let us have two instances d_1, d_2 for which $d_1 = q_1$ and $d_2 = q_2$ and probabilities p_1, p_2 in the $(P-1)$-simplex. Let the vertices in the $(P-1)$-simplex for q_1 and q_2 be given by y_1 and y_2. Let $v_{q_1,q_2} := A_{q_1,\cdot} - A_{q_2,\cdot}$. The orientation function is then defined as

$$O\Big(y_1, y_2, p_1, p_2, v_{q_1,q_2}\Big) := \Big(v_{q_1,q_2} \cdot (y_1 - y_2)\Big)\Big(v_{q_1,q_2} \cdot (p_1 - p_2)\Big) \tag{4.47}$$

What this means is that the orientation function O, with respect to the earlier introduced definition of correct ranking, returns a positive when the estimated probabilities are ranked correctly, a negative when they are ranked incorrectly, and a 0 when they are tied in rank.

To place this in the context of this thesis, let us again have two documents i_1 and i_2, with $d'_{i_1,j} = q_1$ and $d'_{i_2,j} = q_2$ as the true classes. We have estimated probabilities $\widehat{p}_{i_1,j}$ and $\widehat{p}_{i_2,j}$ in the $(P-1)$-simplex and the positions of the true classes in the $(P-1)$-simplex are given by $y_{1,j}$ and $y_{2,j}$. Let A be the partition matrix associated with the classification problem, from which v_{q_1,q_2} can be derived. The orientation function is then given by

$$O\Big(y_{1,j}, y_{2,j}, \widehat{p}_{i_1,j}, \widehat{p}_{i_2,j}, v_{q_1,q_2}\Big) := \Big(v_{q_1,q_2} \cdot \Big(y_{1,j} - y_{2,j}\Big)\Big)\Big(v_{q_1,q_2} \cdot \Big(\widehat{p}_{i_1,j} - \widehat{p}_{i_2,j}\Big)\Big) \tag{4.48}$$

Now, we will use the orientation function O for the construction of a new adapted indicator function. In the binary setting, we introduced the adapted indicator function $\tilde{1}$ so that we could compute the AUC as in Equation 4.42. $\tilde{1}$ returns 1 for a correct ranking, 0 for an incorrect ranking and 0.5 when two estimated probabilities are tied in rank. We aim to construct a similar function in the multinomial setting. Given two instances $\widehat{p}_{i_1,j}, \widehat{p}_{i_2,j}$ and the true class labels and partition matrix, the orientation function gives a positive, negative or 0 output. Thus we define

$$\tilde{1} \circ O\Big(y_{1,j}, y_{2,j}, \widehat{p}_{i_1,j}, \widehat{p}_{i_2,j}, v_{q_1,q_2}\Big) \tag{4.49}$$

as the adapted indicator function that indicates whether two classes are ranked correctly for the multinomial setting. It does this in the same fashion that $\tilde{1}$ does for the binary setting; if a 1 is returned, $\widehat{p}_{i_1,j}, \widehat{p}_{i_2,j}$ are ranked correctly, if a 0 is returned, they are ranked incorrectly and when 0.5 is returned their rank is tied. This all with respect to the decision boundary v_{q_1,q_2}, which the simplex between their true labels.

Now to move towards comparing not just two instances of two different classes, but multiple, all still of either of the two classes. Let $I_{q_1} = \{i : d'_{i,j} = q_1\}$, $I_{q_2} = \{i : d'_{i,j} = q_2\}$, and let $n_{q_1} = |I_{q_1}|, n_{q_2} = |I_{q_2}|$. We introduce the *separability measure*, defined as

$$S(q_1, q_2) := \frac{1}{n_{q_1} n_{q_2}} \sum_{i_{q_1} \in I_{q_1}} \sum_{i_{q_2} \in I_{q_2}} \tilde{1} \circ O\Big(y_{1,j}, y_{2,j}, \widehat{p}_{i_{q_1},j}, \widehat{p}_{i_{q_2},j}, v_{q_1,q_2}\Big) \tag{4.50}$$

Note the similarity between the separability measure and the AUC estimate in Equation 4.42. Both take the average value over an indicator function returning 1, 0 or 0.5 over two sets of classes. In fact, in the binary setting where there are only 2 classes and one works with probability estimates in the 1-simplex, S is equivalent to the AUC estimate as in Equation 4.42. In the multinomial setting, the separability measure still only accounts as a way to compare instances from 2 of the P classes. This brings us to the final step. For the multinomial setting with P classes, we now take the average of S over all possible class combinations and define the AUC_μ as

$$AUC_\mu := \frac{2}{P(P-1)} \sum_{q_1 < q_2} S(q_1, q_2) \tag{4.51}$$

We can thus regard the AUC_μ as taking the average of multiple AUC values.

5

Elasic net models

In chapter 2 we defined our goal of predicting the $d'_{i,j}$, the coarsened driver score for a document i and driver j, through the use of the probabilistic model and computation of $\widehat{p}_{i,j}$. In chapter 3 we saw how to turn a review into meaningful variables $x_1, \ldots, x_V$ and now, in chapter 5, we will see how we can use these variables to compute $\widehat{p}_{i,j}$. We will discuss the so called *Elastic net* model, of which *LASSO* is a special case. We will start with the motivation for this method, explain how the method works and introduce variations of this model.

5.1. Predicting with many variables; motivation for the method

In chapter 3 we arrived at a problem of the following form:

binary_indicator	blunt	close	garbag	good	im	product	shave	...
0	1	1	1	1	1	1	1	...
1	0	0	0	0	0	0	3	...
0	0	0	1	0	0	0	0	...
0	0	0	0	0	0	0	3	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Figure 5.1: Example of the document-term matrix.

The first column contains the variables we want to predict, d'_j, for a driver j in the binary setting and the other columns are predictors $x_1, \ldots, x_V$. In Equations 2.9 and 2.10 we defined how p_j is the underlying probability influencing the Bernoulli outcomes of d'_j, i.e. the values $d'_{i,j}$ with doc_i as input for p_j. Since p_j is unknown, we try to estimate it as $\widehat{p}_j$.

One way of doing this, would be to apply *maximum likelihood estimation*. In doing this, we compute the *likelihood function* over the Bernoulli distribution for all documents i, and then maximize it for a certain choice of parameters. We do not have these parameters yet, but we can introduce them through the use of the *logistic model*, which we will explain in this chapter. Maximum likelihood estimation will give us the choice of parameters that best fits our assumed statistical model, but this will not suffice and we will introduce the *LASSO* which adds a *penalty parameter*. LASSO is a special case of what are called *Elasic net* models. In the example above and this little write up, a binary setting was assumed. We will generalize for the multinomial setting later on in this chapter.

5.1.1. Resembling probability as coefficients

In the logistic model the log odds of the outcomes of a binary variable are modeled as being a linear combination of one or more independent variables. Every variable gets a corresponding coefficient.

Definition 5.1 (Logistic model)
Consider a model with predictor variables $x_1,\dots,x_V$ and a binary response variable Y. We denote $p = P(Y=1)$. A linear relationship is assumed between the predictor variables and the log odds of the event that $Y=1$. Then coefficients $\beta_0,\beta_1,\dots,\beta_V$ are introduced and the linear relationship can be written in the following form:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_V x_V \tag{5.1}$$

Within this thesis, log is considered as the natural logarithm, meaning the logarithm with base e. Other choices for the base of the algorithm are possible for a logistic model, but are not considered in this thesis. β_0 does not correspond to any variable and is named the *intercept*.

In the context of our problem we have a binary variable d'_j and model the log odds of the event $d'_j = 1$ as a linear combination of independent variables, $x_1,\dots,x_V$. For the log odds we will briefly return to how we defined the odds p_j in the first place. From Equation 2.9 we have that

$$p_j(\text{doc}) := P\left(d'_j = 1|\text{doc}\right)$$

Using this, we apply the logistic model

$$\log\left(\frac{p_j}{1-p_j}\right) = \beta_{0,j} + \beta_{1,j} x_1 + \cdots + \beta_{V,j} x_V \tag{5.2}$$

Notice that p_j, and thus by Equation 2.9 the outcome d'_j, depends on the information from the document through the variables $x_1,\dots,x_V$ and that the coefficients $\beta_{0,j},\beta_{1,j},\dots\beta_{V,j}$ are independent from the document and are, in general, different for a different driver j. Now, we rewrite Equation 5.2 and find

$$\frac{p_j}{1-p_j} = \exp\left(\beta_{0,j} + \beta_{1,j} x_1 + \cdots + \beta_{V,j} x_V\right) \tag{5.3}$$

$$\iff \quad p_j = (1-p_j)\exp\left(\beta_{0,j} + \beta_{1,j} x_1 + \cdots + \beta_{V,j} x_V\right) \tag{5.4}$$

$$\iff \quad p_j = \frac{\exp\left(\beta_{0,j} + \beta_{1,j} x_1 + \cdots + \beta_{V,j} x_V\right)}{1+\exp\left(\beta_{0,j} + \beta_{1,j} x_1 + \cdots + \beta_{V,j} x_V\right)} \tag{5.5}$$

$$\iff \quad p_j = \frac{1}{1+\exp\left(-\beta_{0,j} - \beta_{1,j} x_1 - \cdots - \beta_{V,j} x_V\right)} \tag{5.6}$$

$$= P\left(d'_j = 1|\text{doc}\right) \tag{5.7}$$

Now, let $\underline{x}$ and $\underline{\beta}_j$ be vectors of length $V+$ defined as

$$\underline{x} := \begin{bmatrix} x_1 \\ \vdots \\ x_V \end{bmatrix}, \underline{\beta}_j := \begin{bmatrix} \beta_{1,j} \\ \vdots \\ \beta_{V,j} \end{bmatrix} \tag{5.8}$$

Then we have as a short notation for the expression of p_j

$$p_j = \frac{\exp\left(\beta_0 + \underline{\beta}_j^T \underline{x}\right)}{1+\exp\left(\beta_0 + \underline{\beta}_j^T \underline{x}\right)} = \frac{1}{1+\exp\left(-\beta_0 - \underline{\beta}_j^T \underline{x}\right)} \tag{5.9}$$

Thus, we have now explicitly expressed p_j in terms of coefficients $\beta_{0,j},\dots,\beta_{V,j}$ and we can look at estimating these coefficients as a means to estimate p_j.

5.2. Maximizing the likelihood

To estimate the coefficients, we first introduce the likelihood function and maximum likelihood estimator.

Definition 5.2 (The likelihood function and maximum likelihood estimator)
Let $y_1,\dots,y_n$ be observations from n independent and identically distributed random variables drawn from a probability distribution with probability density function f, which is known to be from a family of distributions that depend on parameter $\theta \in \Theta$. Θ is the parameter space, a finite-dimensional Euclidian subspace containing all possible choices for θ. The likelihood function, $\mathscr{L}$ is then defined as

$$\mathscr{L}\left(\theta|y_1,\dots,y_n\right) = f\left(x_1,\dots,x_n|\theta\right) = f\left(x_1|\theta\right)\times f\left(x_2|\theta\right)\times\cdots\times f\left(x_n|\theta\right)$$

The maximum likelihood estimator, $\widehat{\theta}$, is defined as the value for θ that maximizes the likelihood function $\mathscr{L}$, thus

$$\widehat{\theta} = \arg\max_{\theta\in\Theta} \mathscr{L}\left(\theta|y_1,\dots,y_n\right)$$

It is often easier to work with the *log-likelihood function*, which is found by taking the logarithm of $\mathscr{L}$, i.e.

$$\log\mathscr{L}\left(\theta|y_1,\dots,y_n\right) = \sum_{i=1}^{n}\log\Big(f\left(x_i|\theta\right)\Big) \tag{5.10}$$

Since the logarithm is a monotonic function, $\widehat{\theta}$ maximizes $\mathscr{L}$ if and only if $\widehat{\theta}$ maximizes $\log\mathscr{L}$.

Now, we will estimate the coefficients from Equation 5.2 by computing the maximum likelihood estimator. For this, first look at d'_j. Remember that, by Equation 2.10 we have that

$$d'_j|\text{doc} \sim \text{Bern}\Big(p_j(\text{doc})\Big), \qquad \forall\, j\in\{1,\dots,m\}$$

and thus we have for the probability mass function of d'_j, with outcomes in $\{0,1\}$, that

$$f\Big(d'_j|p_j\left(\text{doc}\right)\Big) = p_j\left(\text{doc}\right)^{d'_j}\Big(1-p_j\left(\text{doc}\right)\Big)^{1-d'_j}, \qquad \forall\, j\in\{1,\dots,m\} \tag{5.11}$$

Now to compute the likelihood function. For driver j, take the n pairs of $d'_{i,j}$ and doc_i, fill them in in the probability mass function, and take the product over them to obtain the likelihood function.

$$\mathscr{L}\Big(p_{1,j},\dots,p_{n,j}|d'_{1,j},\dots,d'_{n,j},\text{doc}_1,\dots,\text{doc}_n\Big) := \prod_{i=1}^{n} p_{i,j}^{d'_{i,j}}(1-p_{i,j})^{1-d'_{i,j}}, \quad \forall\, i\in\{1,\dots,n\}, \forall\, j\in\{1,\dots,m\} \tag{5.12}$$

, where we used the notation $p_{i,j}$ as defined in Equation 2.13. In Equation 5.6 we found an explicit expression for p_j in terms of coefficients $\beta_{0,j},\dots,\beta_{V,j}$, and now we want to choose these coefficients such that they maximize $\mathscr{L}$. It is easier to maximize the $\log\mathscr{L}$. For ease of notation, we do not express $p_{i,j}$ in terms of $\beta_{0,j},\dots,\beta_{V,j}$ yet. We get

$$\log\mathscr{L}(p_{1,j},\dots,p_{n,j}|d'_{1,j},\dots,d'_{n,j},\text{doc}_1,\dots,\text{doc}_n) = \log\left(\prod_{i=1}^{n} p_{i,j}^{d'_{i,j}}(1-p_{i,j})^{1-d'_{i,j}}]\right) \tag{5.13}$$

$$= \sum_{i=1}^{n}\log\left(p_{i,j}^{d'_{i,j}}(1-p_{i,j})^{1-d'_{i,j}}\right) \tag{5.14}$$

$$= \sum_{i=1}^{n} d'_{i,j}\log\Big(p_{i,j}\Big) + (1-d'_{i,j})\log\Big(1-p_{i,j}\Big) \tag{5.15}$$

$$= \sum_{i=1}^{n} d'_{i,j}\left(\log\Big(p_{i,j}\Big) - \log\Big(1-p_{i,j}\Big)\right) + \log\Big(1-p_{i,j}\Big) \tag{5.16}$$

$$= \sum_{i=1}^{n} d'_{i,j}\left(\log\left(\frac{p_{i,j}}{1-p_{i,j}}\right)\right) + \log\Big(1-p_{i,j}\Big) \tag{5.17}$$

Now to express this in terms $\beta_{0,j},\dots,\beta_{V,j}$. We already found an expression for $\log\left(\frac{p_{i,j}}{1-p_{i,j}}\right)$ in Equation 5.2. Using Equation 5.5, it is quite easy to express $\log\left(1-p_{i,j}\right)$ as well:

$$\log\left(1-p_{i,j}\right) = \log\left(1-\frac{\exp\left(\beta_{0,j}+\beta_{1,j}x_1+\cdots+\beta_{V,j}x_V\right)}{1+\exp\left(\beta_{0,j}+\beta_{1,j}x_1+\cdots+\beta_{V,j}x_V\right)}\right) \tag{5.18}$$

$$= \log\left(\frac{1}{1+\exp\left(\beta_{0,j}+\beta_{1,j}x_1+\cdots+\beta_{V,j}x_V\right)}\right) \tag{5.19}$$

$$= -\log\left(1+\exp\left(\beta_{0,j}+\beta_{1,j}x_1+\cdots+\beta_{V,j}x_V\right)\right) \tag{5.20}$$

Thus, using Equations 5.2, 5.17 and 5.20 we find

$$\log\mathscr{L}\left(\beta_{0,j},\dots,\beta_{V,j}|d'_{1,j},\dots,d'_{n,j},\mathrm{doc}_1,\dots,\mathrm{doc}_n\right) \tag{5.21}$$

$$= \sum_{i=1}^{n}\left(d'_{i,j}\left(\beta_{0,j}+\beta_{1,j}x_{i,1}+\dots\beta_{V,j}x_{i,V}\right)-\log\left(1+\exp\left(\beta_{0,j}+\beta_{1,j}x_{i,1}+\dots\beta_{V,j}x_{i,V}\right)\right)\right) \tag{5.22}$$

$$= \sum_{i=1}^{n}\left(d'_{i,j}\left(\beta_0+\underline{\beta}_j^T\underline{x}_i\right)-\log\left(1+\exp\left(\beta_0\underline{\beta}_j^T\underline{x}_i\right)\right)\right) \tag{5.23}$$

Similar as to how we defined $\underline{x}$, $\underline{x}_i$ is here defined to be the vector of length V having $x_{i,1},\dots,x_{i,V}$ as its entries. Now, we can finally use this function to compute the maximum likelihood estimate for coefficients $\beta_{0,j},\dots,\beta_{V,j}$. Name these estimates $\widehat{\beta}_{0,j},\dots,\widehat{\beta}_{V,j}$, and in vector form $\underline{\widehat{\beta}}_j$. We say that

$$\underline{\widehat{\beta}}_j := \arg\min_{\beta_0,\underline{\beta}_j\in\mathbb{R}^{V+1}} -\sum_{i=1}^{n}\left(d'_{i,j}\left(\beta_{0,j}+\beta_{1,j}x_{i,1}+\dots\beta_{V,j}x_{i,V}\right)-\log\left(1+\exp\left(\beta_{0,j}+\beta_{1,j}x_{i,1}+\dots\beta_{V,j}x_{i,V}\right)\right)\right) \tag{5.24}$$

$$= \arg\min_{\beta_0,\underline{\beta}_j\in\mathbb{R}^{V+1}} -\sum_{i=1}^{n}\left(d'_{i,j}\left(\beta_0+\underline{\beta}_j^T\underline{x}_i\right)-\log\left(1+\exp\left(\beta_0+\underline{\beta}_j^T\underline{x}_i\right)\right)\right) \tag{5.25}$$

Notice that we take the minimum over the negative of $\log\mathscr{L}$, which is the same as taking the maximum.

We have derived the log likelihood, and found a way of estimating $\underline{\beta}_j$ such that our estimate $\underline{\widehat{\beta}}_j$ is the most probable under the statistical model that we assumed. However, we do need do not account for the problem of *overfitting* and that is why we move to the LASSO.

5.2.1. The penalty term

Through the use of the log-likelihood function we found our estimates $\underline{\widehat{\beta}}_j$ by maximizing the likelihood function, but now these values are overfitted. We only expect a few of the variables $x_1,\dots,x_V$ to possibly influence p_j, and we want to see effect in only a few of the corresponding coefficients $\beta_{1,j},\dots,\beta_{V,j}$. However, since we work with only a finite sample, we will see effects in the estimated coefficients that appear because of sheer coincidence. A larger sample size would help, but the effect of overfitting will persist.

There is a way to combat the problem of overfitting though. We sum a *penalty term* to the log-likelihood term we found earlier. The penalty term will be of the form

$$\lambda||\underline{\beta}_j|| \tag{5.26}$$

$||\cdot||$ being a norm that is yet to be chosen, and λ being a free parameter greater than 0 that represents the amount of regularization desired. Therefore, λ is called the *regularization parameter*. The penalty term is

thus strictly positive. The penalty term then grows with larger coefficients (positive or negative) and decreases when the coefficients are close or equal to 0.

Before summing the penalty term to the log-likelihood function, we first normalize the variables. We do this, because when one variable naturally attains smaller values in our data set, it may cause the corresponding coefficient to grow large, relatively to other coefficients. This will in turn cause the penalty term to grow large. The effect can occur vice versa for a variable that naturally attains rather large values. Its corresponding coefficient could become rather small, with a low impact on the penalty parameter. Of course, we want the penalty to be small in some cases and large in others, but this should be a result of how informative some variables are for our prediction, not of the distributions of these variables in the data. Thus, we normalize all variables in the data set. Let $x_v, v \in \{1,\dots,V\}$ be a variable in our data set, with

$$\overline{x_v} = \frac{1}{n}\sum_{i=1}^{n} x_{i,v} \tag{5.27}$$

$$s = \sqrt{\frac{\sum_{i=1}^{n}\left(x_{i,v} - \overline{x_v}\right)^2}{n}} \tag{5.28}$$

as respectively the sample mean and sample variance. Then, we now redefine $x_{i,v}, \forall\, i \in \{1,\dots,n\}$ as

$$x_{i,v} = \frac{x_{i,v} - \overline{x_v}}{s} \tag{5.29}$$

Repeat this for all variables $x_v, v \in \{1,\dots,V\}$, and we have normalized our predictors. For the rest of this chapter, we assume that variables x_v are normalized. Having done that, we can look at using the penalty term $\lambda||\underline{\beta}_j||$. Summing it to the log-likelihood and minimizing over the whole entity gives us the following result

$$\arg\min_{\beta_0,\underline{\beta}_j \in \mathbb{R}^{V+1}} -\frac{1}{n}\sum_{i=1}^{n}\left(d'_{i,j}\left(\beta_0 + \underline{\beta}_j^T \underline{x}_i\right) - \log\left(1 + \exp\left(\beta_0 + \underline{\beta}_j^T \underline{x}_i\right)\right)\right) + \lambda||\underline{\beta}_j|| \tag{5.30}$$

The estimate, $\widehat{\underline{\beta}}_j$, will now be regularized. This means that, because the use of large positive or negative values is penalized by the penalty parameter, it will avoid these choices, except if their effect in minimizing the log-likelihood term is enough to compensate. Notice that we added $\frac{1}{n}$ as a scale on the log-likelihood because we do not want to have to adapt the penalty parameter λ for the number of documents used to compute the estimate. The λ is for now a free parameter that we can choose. A higher value for λ will lead to a more regularized model, where more coefficients will be similar, and a lower value for λ will lead to more of an (over) fitted model, where more coefficients take seemingly random values.

The use of what norm to use in the penalty term is up for debate. We could use the regular Euclidian norm, also known as the l_2 norm, which is defined as

$$||\underline{\beta}_j||_{l_2} = \sqrt{\sum_{v=1}^{V} \beta_{v,j}^2} \tag{5.31}$$

The use of this form and the resulting minimization problem is known as *Ridge regression*. In this thesis however, we are more interested in using the l_1 norm, which is defined as

$$||\underline{\beta}_j||_{l_1} = \sum_{v=1}^{V} |\beta_{v,j}| \tag{5.32}$$

The resulting minimization problem is known as the *LASSO*. LASSO is an acronym for *Least Absolute Shrinkage and Selection Operator*. The LASSO is a regression analysis method that, besides just regularization, also performs variable selection.[15] Essentially, the use of the l_1 norm in the penalty, makes many of the estimated coefficients shrink to 0 and a higher value for λ will lead to more coefficients shrinking to 0. The coefficients that are retained (those that are not 0) are the ones selected by the LASSO for having predictive value. A lower value for λ will lead to fewer coefficients being estimated as 0. In the context of this thesis, this property can be particularly useful. For a driver j it will select a set of variables (being words) that are important for estimating p_j. Ridge regression does not have this property, although we could think of alternatives.

For example, we could only take those variables for which the estimate for the corresponding coefficient is higher (or lower) than a certain threshold.

Note that we have introduced the λ parameter in the penalty term, and so far not discussed how to use it, only its effect. In the next section, we will discuss how to choose λ.

5.2.2. Choosing the regularization parameter

The λ parameter will be chosen using k-fold cross validation and the AUC. We choose a range of possible values for λ of which we are almost sure that it will contain the 'optimal' value for λ. First we search for the upper bound of this range. As mentioned before, higher values for λ will lead to more coefficients being estimated as 0. We compute the minimal value for λ such that all coefficients are estimated as being equal to 0. Name this value $\lambda_{\max}$. $\lambda_{\max}$ can be found as

$$\lambda_{\max} = \frac{\max_{v\in\{1,\dots,V\}} \sum_{i=1}^{n} x_{i,v} f\left(d'_{i,j}\right)}{n} \tag{5.33}$$

where $f\left(d'_{i,j}\right)$ is the class proportion corresponding to $d'_{i,j}$, i.e. if $d'_{i,j} = 1$ it is the total number of 1s divided by all reviews in the data set (0s and 1s)[7]. That is it for the upper bound $\lambda_{\max}$. The rest of the range within which one should search for the optimal λ is then based on the number of variables and the number of observations in the data set. As a rule of thumb, choose

- If the number of variables is larger than the number of observations, search in the range of $[0.01\lambda_{\max}, \lambda_{\max}]$.
- If the number of observations is larger than the number of variables, search in the range of $[0.0001\lambda_{\max}, \lambda_{\max}]$.

The range within which to search for λ is also chosen in this manner within this thesis. We then choose 100 values for λ in this range. They are chosen such that they divide the log-linear transformation of the range in equal parts. For each λ, we compute the model that is found by the LASSO for that λ and estimate its performance. Performance is estimated using the AUC and k-fold cross validation. The models found for every λ are made using the same training sets and validated on the same test sets. The average is taking over the results from each fold, and we find an estimate of the model performance for each λ. Figure 5.2 shows the series of λ's for two different drivers j.

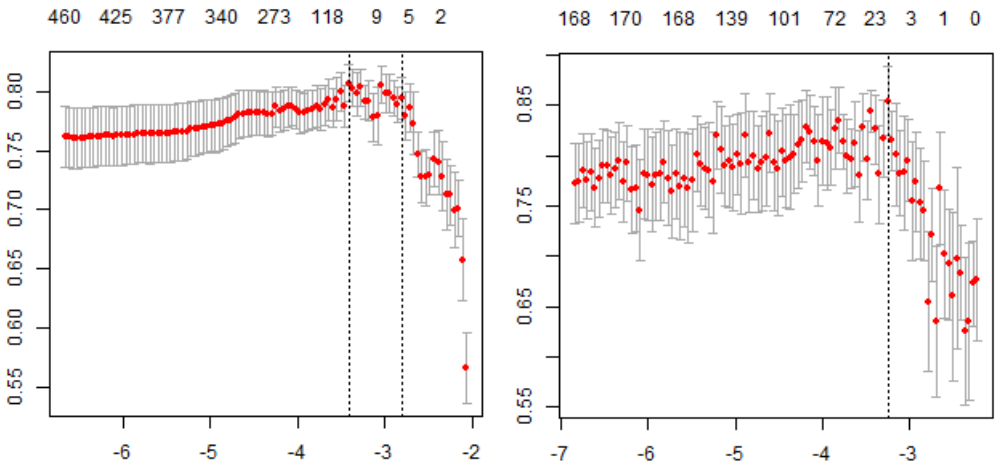


Figure 5.2: Plot of AUC estimate for models found for 100 different values of λ in the LASSO. On the vertical axis the AUC estimate is shown, on the lower axis the value of $\log(\lambda)$. The top axis shows the number of non-zero variables in the model associated with the corresponding λ on the lower axis. The red dots show the values found for λ averaged over the k folds, and the brackets around them give the range of plus one to minus one standard error. A 10-fold cross validation was used. Results were obtained in the binary setting and the model was made to predict positive sentiment for drivers 8 (left picture) and 10 (right picture) in the n775-data set. The variables were of document-term matrix form.

The figure shows the dilemma that arises when it comes to the matter of choosing the value of λ. In the left

picture, the left dotted line shows the location of the λ for the model that gave the highest AUC estimate. The right dotted line gives the location of the most regularized model such that the estimated AUC is still within one standard error from the AUC estimate found by the left dotted line. The standard error is here defined as $\frac{s}{\sqrt{k}}$, where s is the sample variance, as defined in Equation 5.28, of the k AUC estimates we found. One might prefer this 'one standard error lower-model' because it is a more regularized model, and thus less prone to being overfitted, than the model for which we found the highest AUC, but is reasonably close in its performance. So close, that its estimated performance is within one standard error of the estimated performance of the best model. In the right picture, there is no such model, and only the location of the model with the maximum AUC estimate is shown.

5.2.3. Ridge and elastic net options

We already discussed the possibility of applying Ridge Regression earlier. The norm used in Equation 5.30 is then an l_2 norm, as given in Equation 5.31. Unlike the LASSO, Ridge regression does not enforce the irrelevant coefficients in $\beta_{0,j},\ldots,\beta_{V,j}$ to shrink all the way down to 0. For us, this makes Ridge a bit of a less interesting option, as it does not perform variable selection. Nonetheless, it still has predictive power and is thus an option we should consider.

It is also possible to combine the options of Ridge and LASSO regression. This is called *Elastic net*[1] regularization. We split the penalty term of Equation 5.30 in two, part l_1 norm and part l_2 norm. This gives the following minimization problem.

$$\arg\min_{\beta_0,\underline{\beta_j}\in\mathbb{R}^{V+1}} -\frac{1}{n}\sum_{i=1}^{n}\left(d'_{i,j}\left(\beta_0+\underline{\beta_j}^T\underline{x_i}\right)-\log\left(1+\exp\left(\beta_0+\underline{\beta_j}^T\underline{x_i}\right)\right)\right)+\lambda\left(\alpha||\underline{\beta_j}||_{l_1}+(1-\alpha)||\underline{\beta_j}||_{l_2}\right) \tag{5.34}$$

, where α is a parameter within $[0,1]$ that has to be chosen. Compare it with Equation 5.30. Note that the LASSO, only using the l_1 norm corresponds to a special case of Elastic net where $\alpha = 1$. Ridge regression is also a special case of Elastic net, where $\alpha = 0$. Note that like LASSO, Elastic net also performs variable selection, in that it lets many of the parameters shrink to 0.

Choosing the λ parameter is done in the same manner as to how this was done for the LASSO. The only difference is that in determining $\lambda_{\max}$, one has to account for the α parameter. The formula becomes:

$$\lambda_{\max} = \frac{\max_{v\in\{1,\ldots,V\}}\sum_{i=1}^{n} x_{i,v} f\left(d'_{i,j}\right)}{\alpha n} \tag{5.35}$$

5.3. Multinomial setting

So far, we have introduced and motivated the use of the LASSO, Ridge and Elastic net regression options in the binary setting. We now wish to generalize these methods to the multinomial setting.

5.3.1. Multiple sets of coefficients

In the multinomial setting, we assume that we have P classes, and that, as in Equation 2.18, d'_j assumes one of these P classes, say class q, with probability $\left(p_j\right)_q$, such that $\sum_{q=1}^{P}\left(p_j\right)_q = 1$. We will again apply a logistic model, but how do we generalize the logistic model from the binary setting to the multinomial setting?

[1] The elastic net was given its name for its usage as a variable selector, like the LASSO. "It is like a stretchable fishing net that retains 'all the big fish'" [18]

Remember, that in the binary setting the assumption of the logistic model made in Equation 5.2 was

$$\log \frac{p_j}{1-p_j} = \beta_0 + \underline{\beta}_j^T \underline{x} \tag{5.36}$$

$$\iff \log \frac{P\left(d'_j = 1|\text{doc}\right)}{P\left(d'_j = 0|\text{doc}\right)} = \beta_0 + \underline{\beta}_j^T \underline{x} \tag{5.37}$$

as $p_j = P\left(d'_j = 1|\text{doc}\right)$ and $1 - p_j = 1 - P\left(d'_j = 1|\text{doc}\right) = P\left(d'_j = 0|\text{doc}\right)$. Here, the outcome 0 functions as the *pivot outcome* and there is no need to estimate $P\left(d'_j = 0|\text{doc}\right)$ since it follows from p_j by $P\left(d'_j = 0|\text{doc}\right) = 1 - p_j$.

Now, in the multinomial setting d'_j takes values in $\{1,\dots,P\}$ with probabilities $\left(p_j\right)_q$, $q \in \{1,\dots,P\}$. We will run $P-1$ independent binary logistic models, using the outcome P as the pivot outcome. Since $\sum_{q=1}^{P}\left(p_j\right)_q = 1$, there is also no need to estimate $\left(p_j\right)_P$ when we already have estimates for $\left(p_j\right)_1,\dots,\left(p_j\right)_{P-1}$. Thus we aim to regress the outcomes $\{1,\dots,P-1\}$ independently against outcome P. Furthermore, every outcome get its own series of coefficients. This means that we get the following series:

$$\log \frac{\left(p_j\right)_q}{\left(p_j\right)_P} = \left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T \underline{x}, \qquad \forall q \in \{1,\dots,P-1\} \tag{5.38}$$

Where $\underline{\beta}_{q,j}^T$ is a vector of length V defined as

$$\underline{\beta}_{q,j} = \begin{bmatrix} \left(\beta_{1,q}\right)_j \\ \vdots \\ \left(\beta_{V,q}\right)_j \end{bmatrix} \tag{5.39}$$

This makes it that in the multinomial setting $\underline{\beta}_j$ becomes a matrix of size $P \times V$, such that the qth row is the vector $\underline{\beta}_{q,j}$ containing the coefficients for class q, but not the intercepts. Now, exponentiating both sides of Equation 5.38 and moving the probability in the denominator of the fraction to the right side, we get

$$\left(p_j\right)_q = \left(p_j\right)_P \exp\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T \underline{x}\right), \qquad \forall q \in \{1,\dots,P-1\} \tag{5.40}$$

And since we know that summing the probablities over 1 must equal 1, we can the express $\left(p_j\right)_P$:

$$\left(p_j\right)_P = 1 - \sum_{q=1}^{q=P-1}\left(p_j\right)_q = 1 - \sum_{q=1}^{q=P-1}\left(p_j\right)_P \exp\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T \underline{x}\right) \tag{5.41}$$

$$\iff \left(p_j\right)_P = \frac{1}{1 + \sum_{q=1}^{P-1}\exp\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T \underline{x}\right)} \tag{5.42}$$

And now, combining this result with Equation 5.40, we get the following expression for the other probabilities $\left(p_j\right)_q$:

$$\left(p_j\right)_q = \frac{\exp\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T \underline{x}\right)}{1 + \sum_{p=1}^{P-1}\exp\left(\left(\beta_{0,p}\right)_j + \underline{\beta}_{p,j}^T \underline{x}\right)}, \qquad \forall q \in \{1,\dots,P-1\} \tag{5.43}$$

In this expression, we have now found a series of V coefficients for every class, except for class P. In this thesis, we however wish to obtain a form where we find coefficients for every class, including P. We do this in

the following manner: First, introduce a Pth series of V coefficients for class P and let it be the null vector, i.e. $\underline{\beta}_{P,j} = 0$. Also, introduce an intercept for class P and let it be 0, i.e. $\left(\beta_{0,P}\right)_j = 0$. Then, introduce P new series of V coefficients and name these $\underline{\beta}'_{q,j}, q \in \{1,\ldots,P\}$. We let these relate to $\underline{\beta}_{q,j}$ by

$$\underline{\beta}_{q,j} = \underline{\beta}'_{q,j} - \underline{\beta}'_{P,j} \iff \underline{\beta}_{q,j} + \underline{\beta}'_{P,j} = \underline{\beta}'_{q,j}, \qquad \forall q \in \{1,\ldots,P\} \tag{5.44}$$

So that $\underline{\beta}_{P,j} = 0$ as introduced earlier and $\underline{\beta}'_{P,j}$ is generally unequal to 0 (although it can be). Using the same procedure, we also introduce P new intercepts $\left(\beta_{0,q}\right)_j, q \in \{1,\ldots,P\}$, for which we say

$$\left(\beta_{0,q}\right)_j = \left(\beta'_{0,q}\right)_j - \left(\beta'_{0,P}\right)_j \iff \left(\beta_{0,q}\right)_j + \left(\beta'_{0,P}\right)_j = \left(\beta'_{0,q}\right)_j, \qquad \forall q \in \{1,\ldots,P\} \tag{5.45}$$

Using these new series of coefficients and the intercepts, we can rewrite the form from Equation 5.43 as

$$\frac{\exp\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T \underline{x}\right)}{1 + \sum_{p=1}^{P-1} \exp\left(\left(\beta_{0,p}\right)_j + \underline{\beta}_{p,j}^T \underline{x}\right)} = \frac{\exp\left(\left(\beta'_{0,P}\right)_j + \underline{\beta}'^T_{P,j} \underline{x}\right) \exp\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T \underline{x}\right)}{\exp\left(\left(\beta'_{0,P}\right)_j + \underline{\beta}'^T_{P,j} \underline{x}\right) \sum_{p=1}^{P} \exp\left(\left(\beta_{0,p}\right)_j + \underline{\beta}_{p,j}^T \underline{x}\right)} \tag{5.46}$$

$$= \frac{\exp\left(\left(\beta'_{0,q}\right)_j + \underline{\beta}'^T_{q,j} \underline{x}\right)}{\sum_{p=1}^{P} \exp\left(\left(\beta'_{0,p}\right)_j + \underline{\beta}'^T_{p,j} \underline{x}\right)} = \left(p_j\right)_q \tag{5.47}$$

So that we have now obtained a way of expressing $\left(p_j\right)_q, q \in \{1,\ldots,P\}$, such that we have coefficients for all P classes that are not necessarily 0. One could argue that it is in fact the beauty of the form in Equation 5.43 that it shows that there is no need for a set of coefficients for the outcome P, since $\left(p_j\right)_P$ follows from the other entries of p_j. Because in this thesis we deem it more valuable to gain a series of non-zero coefficients for all P classes, we choose the form found in Equation 5.47.

We used the original coefficients $\underline{\beta}_{q,j}$ and intercepts $\left(\beta_{0,q}\right)_j, q \in \{1,\ldots,P\}$ to give rise to new coefficients $\underline{\beta}'_{q,j}$ and intercepts $\left(\beta'_{0,q}\right)_j, q \in \{1,\ldots,P\}$ and obtain the form found in Equation 5.47. In the remainder of this chapter, we no longer use the original coefficients and write

$$\left(p_j\right)_q = \frac{\exp\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T \underline{x}\right)}{\sum_{p=1}^{P} \exp\left(\left(\beta_{0,p}\right)_j + \underline{\beta}_{p,j}^T \underline{x}\right)} \tag{5.48}$$

So that $\underline{\beta}_{q,j}$ and $\left(\beta_{0,q}\right)_j, q \in \{1,\ldots,P\}$ now denote the intercepts and coefficients such that $\underline{\beta}_{P,j}, \left(\beta_{0,P}\right)_j$ are not necessarily 0.

5.3.2. Maximizing the likelihood

Now to derive the log likelihood for the multinomial setting. In the multinomial setting, d'_j is a draw from the Categorical distribution with parameter $p_j(\text{doc}) = [\left(p_j\right)_1(\text{doc}), \ldots, \left(p_j\right)_P(\text{doc})]$. The corresponding probability mass function is

$$f\left(d'_j, p_j(\text{doc})\right) = \prod_{q=1}^{P} \left(\left(p_j\right)_q(\text{doc})\right)^{\mathbf{1}_{d'_j = q}} \tag{5.49}$$

Using this, we can compute the likelihood function. For every driver j, we have n pairs of doc_i and $d'_{i,j}$. Thus, we take the product over the likelihood functions with our entered data and find

$$\mathscr{L}(p_{1,j},\ldots,p_{n,j} | d'_{1,j},\ldots,d'_{n,j}, \text{doc}_1,\ldots,\text{doc}_n) = \prod_{i=1}^{n} \prod_{q=1}^{P} \left(\left(p_{i,j}\right)_q\right)^{\mathbf{1}_{d'_{i,j} = q}} \tag{5.50}$$

Now, take the expression for $\left(p_j\right)_q$ found in Equation 5.48 and place it in the expression for $\mathscr{L}$.

$$\mathscr{L}(\underline{\beta}_{1,j},\ldots,\underline{\beta}_{P,j}|d'_{1,j},\ldots,d'_{n,j},\text{doc}_1,\ldots,\text{doc}_n) = \prod_{i=1}^{n}\prod_{q=1}^{P}\left(\frac{\exp\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T\underline{x}\right)}{\sum_{p=1}^{P}\exp\left(\left(\beta_{0,p}\right)_j + \underline{\beta}_{p,j}^T\underline{x}\right)}\right)^{\mathbf{1}_{d'_{i,j}=q}} \tag{5.51}$$

Now, we take the log over it to find the $\log\mathscr{L}$. We derive

$$\log\mathscr{L}(\underline{\beta}_{1,j},\ldots,\underline{\beta}_{P,j}|d'_{1,j},\ldots,d'_{n,j},\text{doc}_1,\ldots,\text{doc}_n) = \log\left(\prod_{i=1}^{n}\prod_{q=1}^{P}\left(\frac{\exp\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T\underline{x}\right)}{\sum_{p=1}^{P}\exp\left(\left(\beta_{0,p}\right)_j + \underline{\beta}_{p,j}^T\underline{x}\right)}\right)^{\mathbf{1}_{d'_{i,j}=q}}\right) \tag{5.52}$$

$$= \sum_{i=1}^{n}\sum_{q=1}^{P}\mathbf{1}_{d'_{i,j}=q}\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T\underline{x_i} - \log\left(\sum_{p=1}^{P}\exp\left(\left(\beta_{0,p}\right)_j + \underline{\beta}_{p,j}^T\underline{x_i}\right)\right)\right) \tag{5.53}$$

$$= \sum_{i=1}^{n}\left(\sum_{q=1}^{P}\mathbf{1}_{d'_{i,j}=q}\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T\underline{x_i}\right) - \sum_{q=1}^{P}\mathbf{1}_{d'_{i,j}=q}\log\left(\sum_{p=1}^{P}\exp\left(\left(\beta_{0,p}\right)_j + \underline{\beta}_{p,j}^T\underline{x_i}\right)\right)\right) \tag{5.54}$$

$$= \sum_{i=1}^{n}\left(\sum_{q=1}^{P}\mathbf{1}_{d'_{i,j}=q}\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T\underline{x_i}\right) - \log\left(\sum_{q=1}^{P}\exp\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T\underline{x_i}\right)\right)\right) \tag{5.55}$$

In the last step, we used that $\sum_{q=1}^{P}\mathbf{1}_{d'_{i,j}=q} = 1$. Now that we have derived the log likelihood for the multinomial setting, we can add a penalty parameter and find expressions for the LASSO and Elastic Net and Ridge Regression options.

5.3.3. Penalizing and Elastic net models

Now how do we choose the penalty parameter in the multinomial setting? Remember that for the binary setting, for the LASSO we chose in Equation 5.32 a penalty parameter that sums the absolute values of the entries of β_j. We wish to do the same in the multinomial setting, but $\underline{\beta_j}$ is a matrix now. Thus, we have

$$||\underline{\beta_j}||_{l_1} = \sum_{q=1}^{P}||\underline{\beta}_{q,j}||_{l_1} = \sum_{q=1}^{P}\sum_{v=1}^{V}|\left(\beta_{v,q}\right)_j| \tag{5.56}$$

for the multinomial setting. The formula giving estimates $\underline{\widehat{\beta}}_j$ for $\underline{\beta_j}$ using the LASSO then becomes

$$\underline{\widehat{\beta}}_j := \arg\min_{\underline{\beta_j}\in\mathbb{R}^{P\times V+1}} -\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{q=1}^{P}\mathbf{1}_{d'_{i,j}=q}\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T\underline{x_i}\right) - \log\left(\sum_{q=1}^{P}\exp\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T\underline{x_i}\right)\right)\right) + \lambda||\underline{\beta_j}||_{l_1} \tag{5.57}$$

Using this technique, the LASSO will perform variable selection, in the multinomial setting, which it also did in the binary setting. Now for the Ridge regression and Elastic net options, we need the l_2-norm for the matrix form of $\underline{\beta_j}$. We set

$$||\underline{\beta_j}||_{l_2} = \sum_{q=1}^{P}||\underline{\beta}_{q,j}||_{l_2} = \sum_{q=1}^{P}\sum_{v=1}^{V}\left(\left(\beta_{v,q}\right)_j\right)^2 \tag{5.58}$$

The formula for estimating $\underline{\beta_j}$ using elastic net then becomes

$$\underline{\widehat{\beta}}_j := \arg\min_{\underline{\beta_j}\in\mathbb{R}^{P\times V+1}} -\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{q=1}^{P}\mathbf{1}_{d'_{i,j}=q}\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T\underline{x_i}\right) - \log\left(\sum_{q=1}^{P}\exp\left(\left(\beta_{0,q}\right)_j + \underline{\beta}_{q,j}^T\underline{x_i}\right)\right)\right) + \lambda\left(\alpha||\underline{\beta_j}||_{l_1} + (1-\alpha)||\underline{\beta_j}||_{l_2}\right) \tag{5.59}$$

For $\alpha \in [0,1]$, such that $\alpha = 1$ corresponds to a LASSO and $\alpha = 0$ corresponds to Ridge regression.

Choosing the value for λ is done in the same manner as in the binary setting, as discussed in 5.2.2. The range in which to search for values for λ is chosen in the exact same way, comparing the number of observations to the number of variables, and the values are then chosen such that they divide the log-linear transformation of this range in equal parts. For every value of λ the corresponding model is generated, and evaluated. The difference between the binary setting and multinomial setting is here. In the binary setting we used the AUC, and in the multinomial setting we use the AUC_μ, as described in Equation 4.2.5. We can get then obtain the same plots as shown in Figure 5.2, but for the multinomial setting. Examples are shown in Figure 5.3.

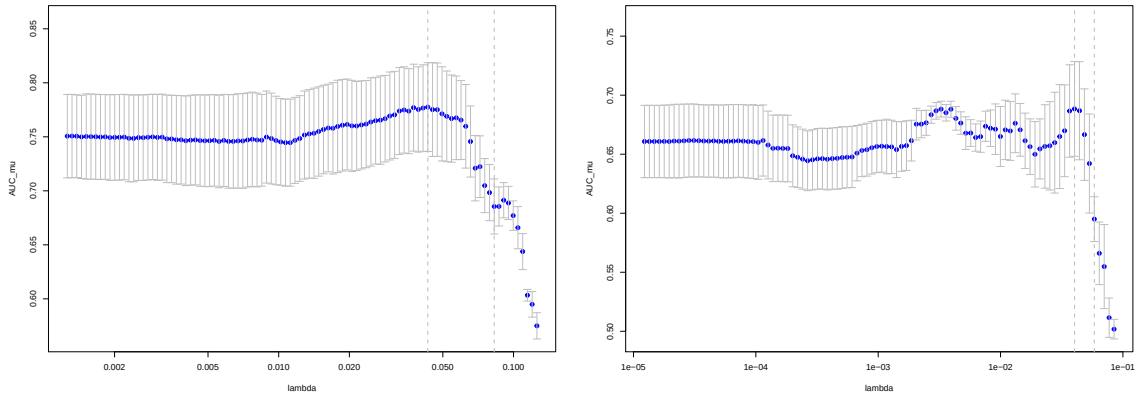


Figure 5.3: Plot of AUC_μ estimates for models found for 100 different values of λ using the LASSO. On the vertical axis the AUC_μ estimate is shown, on the lower axis the value of $\log(\lambda)$. The top axis shows the number of non-zero variables in the model associated with the corresponding λ on the lower axis. The blue dots show the values found for λ averaged over the k folds, and the brackets around them give the range of plus one to minus one standard error. A 5-fold cross validation was used. Results were obtained in the binary setting and the model was made to predict for drivers 6 (left picture) and 9 (right picture) in the n775-data set. The variables were of document-term matrix form.

6

Random forests

In chapter 5 we discussed how we apply Elastic net regression, including LASSO and Ridge regression, as possible methods to solve our problem of estimating $\widehat{d}_j$. In this chapter, we will introduce a different method called *random forests*. In this thesis random forests are used as a classification technique that operates by creating a multitude of *decision trees* and giving the mode of the classes predicted by the trees as output. In this chapter, we will discuss how these random forests work and how we will apply them in the context of this thesis.

6.1. Decision trees

In this section we will give an example and the definition of a decision tree, or rather *classification tree* in the context of this thesis. A classification tree is a type of decision tree, specifically for the application in a classification problem, as is the case in this thesis. When creating a decision tree we aim at creating a flowchart-like structure which can take the attributes of an object, in our case the values $x_{i,1},\ldots,x_{i,V}$ for a document i with the aim of predicting a certain outcome, $d'_{i,j}$ in this case.

To explain the concept of a classification tree, let us look at a simplified example of what could be our data set, and create a decision tree model that could describe this data set.

binary_indicator	blunt	close	garbag	good
0	1	1	1	1
1	1	1	2	0
0	1	0	1	0
1	0	1	0	0

Figure 6.1: Example of the document-term matrix, for a data set with $n = 4$ documents and $V = 4$ unique words.

Based on this data set, we can construct a decision tree model. Many different decision trees are possible. One possibility is presented here:

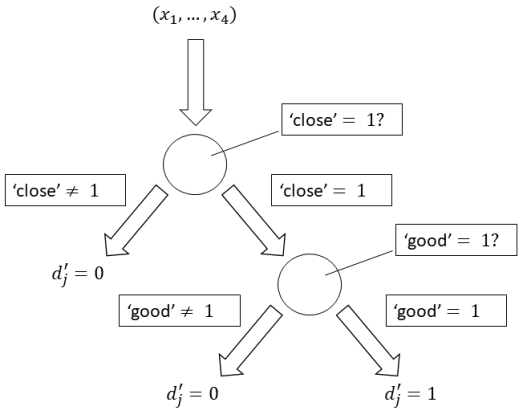


Figure 6.2: Possible decision tree based on the example from 6.1. Variables are $x_1, \ldots, x_4$, where variable x_2 is named 'close' and variable x_4 is named 'good', the predictions are made for d'_j.

The two places where a choice based on the value of the variables for 'close' and 'good' is made are called *nodes*. In this decision tree, two *branches* exit from every node. The decision tree in this example ends in 3 *leaves*, the places where either $d'_j = 0$ or $d'_j = 1$.

Note that not all variables from the example were used in this decision tree. Not all variables need to be used in a decision tree, nor is a decision tree restricted in the number of variables used. We will now explain how we use data to *grow* decision trees for the construction of a *Random forest model*.

6.1.1. Growing the decision tree

First, we need to introduce a new concept.

Definition 6.1 (Gini impurity)
Let $\{y_i\}, i \in \{1, \ldots, n\}$ *be a set of values such that* $y_i \in \{1, \ldots, P\}, \forall i \in \{1, \ldots, n\}$ *where* P *is the number of distinct classes. Let* q *be a class, i.e.* $q \in \{1, \ldots, P\}$. *Define*

$$p_q := \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{y_i = q}$$

So that p_q *is the fraction of* y_i *that are labeled as class* q. *The Gini impurity, GI, of the set* $\{y_i\}$ *is then given by*

$$GI\big(\{y_i : i \in \{1, \ldots, n\}\}\big) = 1 - \sum_{q=1}^{P} p_q^2$$

Let us have data of the form $\left(y, \underline{x}\right)$ where $y \in \{1, \ldots, P\}$ such that P is the number of classes, and $x = \left[x_1, \ldots, x_M\right]$ being a vector of M variables. Assume we have a data set $I = \left\{\left(y_i, \underline{x}_i\right), i \in \{1, \ldots, n\}\right\}$.

Given this, we we will construct the nodes. The first node will partition I in two sets, say I_l and I_r, with sizes n_l and n_r. Name the first node S, and say that the partition is made based on variable $x_m, m \in \{1, \ldots, M\}$. Let s then be an indicator function for which we say that if $s\left(x_{i,m}\right) = 1$, we have $(y_i, \underline{x}_i) \in I_r$ and if $s\left(x_{i,m}\right) = 0$, we have $(y_i, \underline{x}_i) \in I_l$, $\forall i \in \{1, \ldots, n\}$, i.e.

$$I_l := \left\{\left(y_i, \underline{x}_i\right), i \in \{1, \ldots, n\} : s\left(x_{i,m}\right) = 0\right\} \tag{6.1}$$

$$I_r := \left\{\left(y_i, \underline{x}_i\right), i \in \{1, \ldots, n\} : s\left(x_{i,m}\right) = 1\right\} \tag{6.2}$$

We use $S(I) = (I_l, I_r)$ as notation for the partition of I as a result of passing through node S. I_l and I_r are again partitioned by other nodes, say S_l and S_r, and the process starts over again. The following figure gives a brief overview.

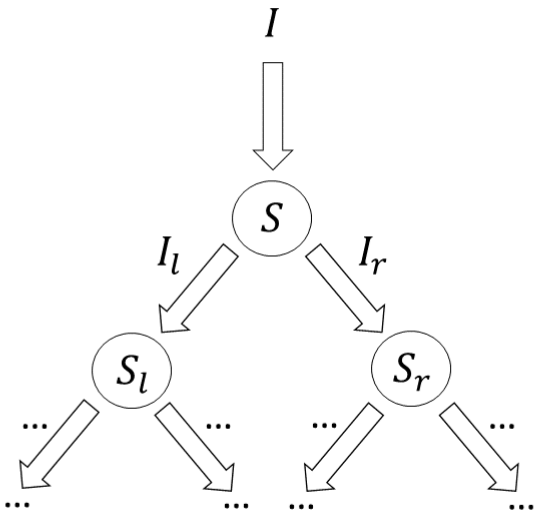


Figure 6.3: Figure representation of growing a decision tree.

When growing a decision tree, we do not choose the variable x_m and function s at random of course. We use the Gini impurity to choose these. x_m is to be chosen among the M variables. We restrict s to a certain set of possible indicator functions. These are relatively simple functions. In the decision tree given in Figure 6.2 we saw two examples of indicator functions. These functions checked whether the incoming variable, in the first node 'close' and in the second node 'good', equalled 1 or not. Another common choice for the indicator functions is one that checks whether a variable is greater than a certain fixed value or not. Then, for all possible choices of x_m and s we can compute I_l and I_r, the sets we have after the partition in node S. We compute the Gini impurity for I_l and I_r and we aim to have a lower Gini impurity after node S than before. With this goal in my mind, we weigh the Gini impurities of the sets according to the the set sizes, n_r and n_l. This means that we have

$$\mathrm{GI}\left(\left\{y_i : \left(y_i, \underline{x}_i\right) \in I\right\}\right) \tag{6.3}$$

as the Gini impurity before node S. After node S, we say

$$\mathrm{WGI}\left(\left\{y_i : \left(y_i, \underline{x}_i\right) \in I_l\right\}, \left\{y_i : \left(y_i, \underline{x}_i\right) \in I_r\right\}\right) = \left(\frac{n_l}{n}\right)\mathrm{GI}\left(\left\{y_i : \left(y_i, \underline{x}_i\right) \in I_l\right\}\right) + \left(\frac{n_r}{n}\right)\mathrm{GI}\left(\left\{y_i : \left(y_i, \underline{x}_i\right) \in I_r\right\}\right) \tag{6.4}$$

where WGI stands for Weighted Gini impurity. We choose x_m and s such that the Gini impurity before S, given by 6.3, minus the Gini impurity after S, given by 6.4, is minimal.

We have established how to choose x_m and s for the first node of the tree. For all other nodes, just repeat this process iteratively. For example for S_l, take I_l as I and repeat the process above. This can be done continuously until we reach the leafs. The leafs are reached, when no partition on the data can be made such that the partition gives a decrease in Gini impurity in every leaf we have sets for which all y_i are the same class. This can happen when all instances in a set are of the same class (as the Gini impurity than equals 0), but this is certainly not always the case. The data can also be composed such that, with the given variables, no

more splits leading to a decrease in Gini impurity can be made, despite the fact that there are still instances of different classes in the leaf. Having established how to grow a tree, we can move on to the random forest model.

6.2. Random forests

The idea of 'Random decision forests' was first proposed in 1995[6], but in 2001 Breiman introduced random forests proper [1]. Random forests operate by constructing a multitude of decision trees and, when used for classification, then taking the mode of the classes predicted by the individual trees. Breiman also describes random forests for regression, with a continuous output, which are obtained by taking the mean prediction of the individual trees. Furthermore, Breiman describes *out of bag* error estimates and *variable importance* measures. In this section, we will describe the random forests as used for classification problems and how we make use of aforementioned measures in the context of our problem.

Definition 6.2 (Random Forest)
Let us have data of the form $\left(y, \underline{x}\right)$ *where* $y \in \{1, \ldots, P\}$ *such that* P *is the number of classes, and* $\underline{x} = \left[x_1, \ldots, x_M\right]$ *being a vector of* M *variables. Assume we have a data set* $I = \left\{\left(y_i, \underline{x}_i\right), i \in \{1, \ldots, n\}\right\}$.

Let $0 < M_D < M, M_D \in \mathbf{N}$. *In the random forest, we will grow a number of decision trees using the Gini impurity (this methods was described in the previous section). Name this number* B. *For every* $b \in \{1, \ldots, B\}$, *sample* M_D *variables from* $\underline{x}$, *without replacement. Name this sample* $\underline{x}_b$. *Also, for every* b *draw* n *samples from* I, *with replacement. We name these samples the bootstrap samples. Name the bootstrap samples* I_b *and let* $\{y_i*, \underline{x}_i*\}, i \in \{1, \ldots, n\}$ *be the elements of* I_b. *Now, let* D_b *be the decision tree grown using the set* I_b, *with only* M_D *variables from* $\underline{x}_b$.

Now, we have B *decision trees in our random forest model. Let* $\underline{x}_{n+1}$ *be a new instance coming in, for which* y_{n+1} *is to be predicted.* $\underline{x}_{n+1}$ *is then sent through every decision tree* D_b *in the forest. The random forest then takes the mode of the predictions of all the individual trees as the outcome of the entire model. In this way, the decision trees 'vote' for the final prediction of the random forest.*

The result of sampling n times with replacement, is that about $2/3$ of the unique instances in the data set are fed to each decision tree. The other $1/3$ of the instances are duplicates. This can be shown with a small calculation. Let $N = \{1, \ldots, n\}$ be a collection from which we draw n times with replacement. Let Y be the number of unique elements drawn from N, and let X_i be a variable for $i \in N$ such that $X_i = 1$ when i is drawn at least once, and $X_i = 0$ when i is not drawn. Then

$$\mathbb{E}[X_i] = P\left(X_i = 1\right) = 1 - P\left(X_i = 0\right) \tag{6.5}$$

Note that when drawing n times $P\left(X_i = 0\right) = \left(\frac{n-1}{n}\right)^n$. Then, to find the expectation of Y, take the sum over all $i \in N$ of the expectation of X_i.

$$\mathbb{E}[Y] = \mathbb{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbb{E}[X_i] = \sum_{i=1}^{n} 1 - P\left(X_i = 0\right) = \sum_{i=1}^{n} 1 - \left(\frac{n-1}{n}\right)^n = n - \frac{(n-1)^n}{n^{n-1}} \tag{6.6}$$

To find the proportion of unique $i \in N$ drawn, divide by n. We find that as n goes to infinity, we have

$$\frac{\mathbb{E}[Y]}{n} = 1 - \frac{(n-1)^n}{n^n} = 1 - \left(1 - \frac{1}{n}\right)^n \to 1 - \frac{1}{e} = 0.6321\ldots. \tag{6.7}$$

In the last step, we used that

$$\lim_{n\to\infty}\left(1 - \frac{1}{n}\right)^n = \lim_{n\to\infty}\left(1 + \frac{-1}{n}\right)^n = e^{-1} = \frac{1}{e} \tag{6.8}$$

Of course, n does not go to infinity in our context, but n is sufficiently large that the proportion of unique instances drawn will be about $2/3$.

Given a data set and our desire to construct a random forest model as described in Definition 6.2, we have two parameters that we still need to choose: M_D and B. For M_D, it is generally recommended to choose $M_D = \left\lfloor \sqrt{M} \right\rfloor$ [17]. Within this thesis, we will also choose M_D in this manner.

For choosing B we need to introduce the concept of *out-of-bag samples* and *out-of-bag error estimate*. We showed that every decision tree is grown using about 2/3 of the unique instances in the data set. The other 1/3 of the instances are called the out-of-bag (OOB) samples. To compute the out-of-bag error estimate for a random forest, first compute the prediction for every instance in the data set using only those decision trees where the instance was OOB in construction. Naturally, this is in about 1/3 of the trees. This prediction is found by sending the instance down these trees, and then taking the mode of the found predictions. Then, doing this for every instance and matching the prediction against its true class, gives the OOB error estimate. The OOB error estimate can be computed using only the first few decision trees in the forest, then one more, another one and so forth. This gives the following picture.

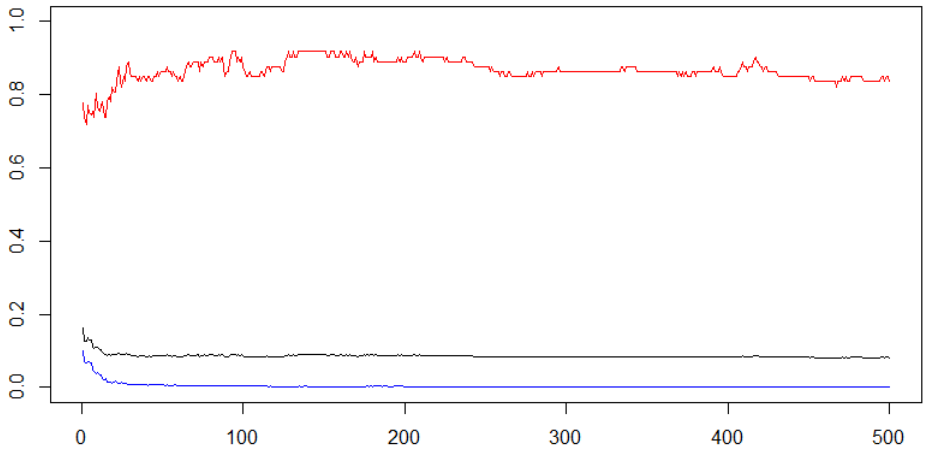


Figure 6.4: Plot of the OOB error rate for a random forest in the binary setting. In red the OOB error rate for the 1s, in blue the 0s and in black the overall OOB error rate. On the y-axis is the OOB error rate, on the x-axis the number of trees. Results were obtained running a random forest with 500 trees for present sentiment for driver number 12, "shaving irritation", in the n775-data set. The indicator matrix form was used as input. The word count threshold was set to 11.

In this case, we ran a random forest model for driver number 12 in the n775-data set with present sentiment in the binary setting, so all instances are labeled as class 1 or class 0. In the plot, the OOB error rate for all instances, i.e. the overall error rate, is plotted in black. We have also plotted in red the OOB error rate for the instances with 1 for its true class, and in blue the OOB error rate for the instances with 0 for its true class. For driver number 12 with present sentiment, we have 73 1s and 702 0s in the data set, so there is a large imbalance. This explains why the black and blue line (for overall error rate and 0s error rate respectively) are so similar in shape. Furthermore, the error rate for the 0s is low (close to 0) and the error rate for the 1s is high (around 0.9).

The OOB error rate can be used as an indication of the accuracy of the model. Furthermore, the plot of the OOB error rate versus the number of trees is used in literature as a way to determine B[17]. The number of trees where the error rate stabilizes is used as an indication of how many trees to grow in the random forest. An indication, because it is generally safe to grow more trees. There is no real downside to growing more trees, aside from the additional computational cost. The random forest does not suffer from overfitting when growing more trees. The error rate stays stable. However, it seems that the behaviour of the OOB error rate is somewhat strange here, and the imbalance of our data set is likely the cause of this. We have a model predicting almost only 0s, and thus the OOB error rate grows stable rather rapidly. It does not seem fair to base our choice for B, the number of trees to grow on this. In the rest of this section, we will show how to overcome this imbalance in our data. We will offer 2 adaptions of the standard random forest method and explain how they overcome the problem of class imbalance. The first adaption is relatively simple and relies on adapting the threshold of the fraction of votes needed for an instance to be predicted as the minority class. The second adaption is somewhat more advanced, and tries to overcome the class imbalance by feeding stratified samples instead of random samples to the decision trees.

6.2.1. Adapting the random forest for class imbalance

In Figure 6.4 we saw that the OOB error rate for instances with 0 for their true class is low and for instances with a 1 as their true class it is rather high. Why does this happen? Every decision tree in the forest is trained on a random bootstrap sample from the data set. Since the class distribution in the data set is imbalanced, this will generally also be the case for the bootstrap samples used to grow the decision trees. When a decision tree is trained on imbalanced data, it will also predict the majority class (0) more often than not, because it has been trained on more instances from this class than of the minority class (1). Let us assume you are looking at an OOB sample with true class 1. Then, when taking the mode of the predictions of all decision trees where the instance was OOB, it is still unlikely that the prediction is a 1, unless the instances that have 1 for its true class are perfectly separable from the instances with true class 0. This is generally not the case though, and also not in this thesis. Thus, the predictions for the OOB samples will far more often be a 0 than a 1. This explains the large difference between the red and blue line in Figure 6.4.

This problem does not only occur when computing the OOB error rate. If we were to use the random forest to predict a new instance, the same effect will occur. More often than not, a majority of the decision trees will predict a 0, and thus the random forest will predict a 0, also when the instance is in fact a 1. So far we have discussed the problem of imbalance for random forests only for the binary setting. The problem also occurs in the multinomial setting, where (in the context of this thesis) the NA class dominates the P and N classes. We aim to overcome this problem by not only predicting the minority class (1 in the binary setting, P or N in the multinomial setting) when a majority of the decision trees predict this class, but already when just a few do this.

Go back to the problem setting of this thesis. For a driver j, we aim to predict d'_j through the use of variables $x_1,\dots,x_V$, where d'_j is a class that takes values in $\{1,\dots,P\}$. Let $\underline{x} = \left[x_1,\dots,x_V\right]$ be the vector of length V with the variables as entries. For every driver j, we then have a data set $\left\{\left(d'_{i,j},\text{doc}_i\right), i \in \{1,\dots,n\}\right\}$ or the equivalent $\left\{\left(d'_{i,j},\underline{x}_i\right), i \in \{1,\dots,n\}\right\}$. Now to use this data for the construction of a random forest.

In Chapter 2 we described how $d'_{i,j}$ is assumed to be the outcome of a stochastic function. We assume that there is an underlying probability p_j giving the distribution from which d'_j is drawn. We aim to estimate p_j with $\widehat{p}_j$. In Chapter 5 we did this by estimating the coefficients created by the assumption of the logistic model. However, we have not introduced any probability p in the use of random forests. The estimate $\widehat{d}'_{i,j}$ is found sending $x_{i,1},\dots,x_{i,V}$ through the decision tree, and the class that wins the majority of the votes becomes the prediction. This is where we now make a change.

Let $\widehat{p}_j$ be the proportion of votes cast by the decision trees. This means that we count the number of votes cast for each of the classes and divide them by B, the number of trees in the random forest. In the binary setting, $\widehat{p}_j$ is only the fraction of votes cast for class 1. In the multinomial setting, $\widehat{p}_j$ is a vector of length P. This is in line with how we introduced $\widehat{p}_j$ in Chapter 2. Formally we write: let RF be a random forest constructed using $\left\{\left(d'_{i,j},\text{doc}_i\right), i \in \{1,\dots,n\}\right\}$ as the training data and B as the number of decision trees grown. Then $\widehat{p}_j\left(\text{doc}\right) = \left[\widehat{p}_1\left(\text{doc}\right),\dots,\widehat{p}_P\left(\text{doc}\right)\right]$ is our estimate for p_j, where $\widehat{p}_q\left(\text{doc}\right)$ is the number of decision trees predicting $\widehat{d}'_j = q$ divided by B when doc is given as input. In the binary setting, we let $\widehat{p}_j\left(\text{doc}\right)$ be the number of decision trees predicting $\widehat{d}'_j = 1$ divided by B. We will also call this the proportion or fraction of votes.

Then, say that we have a new document, e.g. $n+1$, coming in and we have trained our random forest for driver j on the first n documents. In the binary setting, we then predict $d'_{n+1,j}$ as

$$\widehat{d}'_{n+1,j} = \begin{cases} 1, & \widehat{p}_{n+1,j} \geq t \\ 0, & \widehat{p}_{n+1,j} < t \end{cases} \tag{6.9}$$

where t is a threshold value that is to be set. In the random forest method as introduced in Definition 6.2 $\widehat{d}'_{n+1,j}$ was determined through the majority of the votes of the decision trees. This corresponds to $t = 0.5$. In the multinomial setting, $\widehat{d}'_{n+1,j}$ is determined from $\widehat{p}_{n+1,j}$ through the use of a misclassification matrix A, as described in Chapter 2. The misclassification matrix imposes a partition on the $\left(P-1\right)$-simplex. Choosing A

as a matrix imposing equal cost on all misclassifications corresponds to taking the mode of the votes of the decision trees, as is done in the random forest method described in Definition 6.2.

This establishes how we compute $\widehat{p}_j$ in both the binary and the multinomial setting and how we can adapt the threshold value (in the binary setting) or misclassification matrix (in the multinomial setting) to influence a prediction $\widehat{d}_j$ given by the random forest. In the binary setting, we want to lower the threshold to compensate for the imbalance between the 1s and 0s. A possible choice for t is the total number of 1s in the data set, divided by the total number of reviews in the data set, 0s and 1s. Using the example introduced in Figure 6.4, this gives $t \approx 0.10$. This same threshold value can be used when calculating the OOB error rate. Choosing this threshold than means that, as only than 10% of the votes from the OOB decision trees are needed to predict a 1, almost 90% of the votes from the OOB decision trees are needed to predict a 0. Let us plot the OOB error rates against the number of trees again, but now with the lowered threshold value.

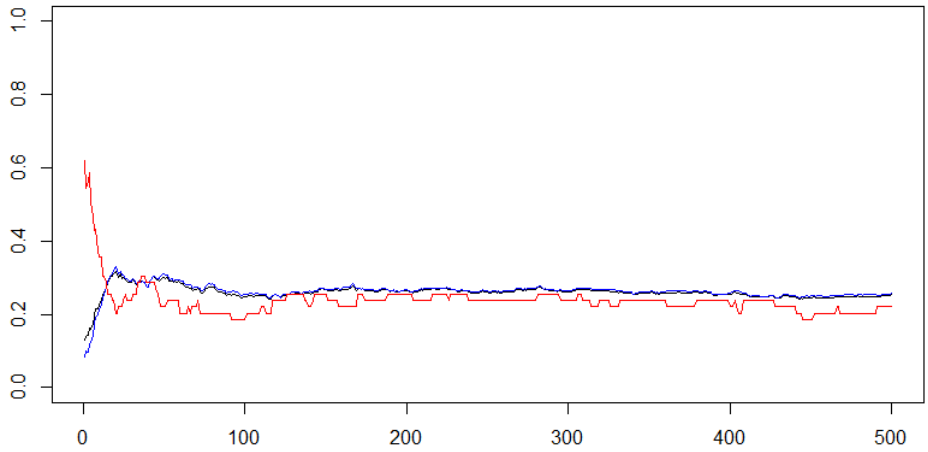
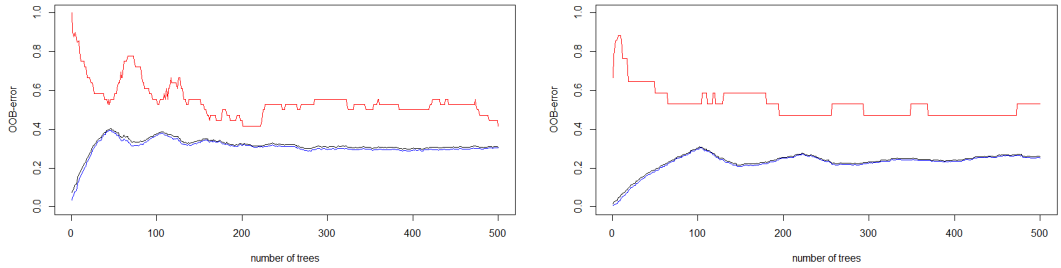


Figure 6.5: Plot of the OOB error rate for a random forest in the binary setting, where the threshold value t was set equal to the fraction of instances labeled as 1 (as opposed to 0). In red the OOB error rate for the 1s, in blue the 0s and in black the overall OOB error rate. On the y-axis is the OOB error rate, on the x-axis the number of trees. Results were obtained running a random forest with 500 trees for present sentiment for driver number 12, "shaving irritation", in the n775-data set. The indicator matrix form was used as input. The word count threshold was set to 11.

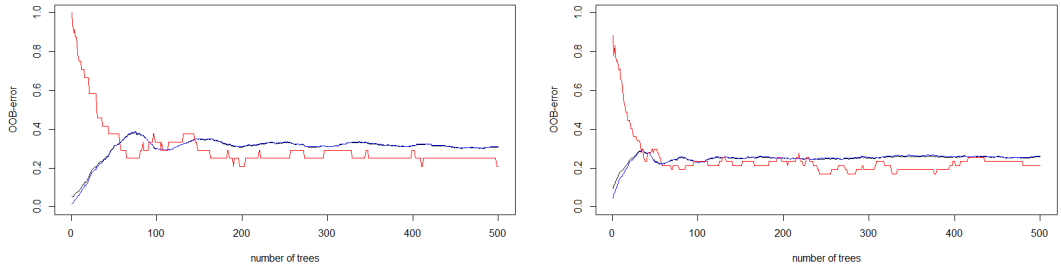
We see that the OOB error rate for both the instances whose true class is 1 and the instances whose true class is 0 are now around 0.25. The error rate also stabilises at a higher number of trees. This is caused by the random forest not predicting almost exclusively 0s anymore. Furthermore, this has also caused an increase in the overall OOB error rate. The OOB error rate is also an indicator for the accuracy of the model, so this increase does not seem like a good thing. In this thesis we are however not too interested in the model with the highest accuracy, if that means that it predicts (almost) only 0s. We are interested in finding a model that is able to distinguish 1s from the 0s. That is, for an instance i we are interested in the relation between $\widehat{p}_{i,j}$ and $d'_{i,j}$, not so much $\widehat{d}'_{i,j}$. To this end, we compute the AUC (in the binary setting) and the AUC$_\mu$ (in the multinomial setting) for the models. In Chapter 7 we will discuss how we compute these for random forests using a cross validation. By Equations 4.42 and 4.51 in Chapter 4 it is clear though that we need the true labels, i.e. d'_j, and the probability estimates $\widehat{p}_j$ for this. The true labels we have in our data set and in the previous part of this chapter we have established how we can find $\widehat{p}_j$.

We have stated our desire to choose B such that the OOB error rate is stable. Earlier, we chose t as the fraction of 1s in the data set, looked at the effect this choice had on the development of the OOB error rate as the number of trees increased. Let us take a look at a few more plots of the OOB error rate, to get an idea of what number of trees to grow.



(a) driver 13, positive sentiment, mean matrix form, word count threshold 21.

(b) driver 3, negative sentiment, normalized tf-idf matrix form, word count threshold 6.



(c) driver 8, negative sentiment, document term matrix form, word count threshold 16.

(d) driver 15, present sentiment, indicator matrix form, word count threshold 11.

Figure 6.6: Plots of the OOB error rate for a random forest in the binary setting, where the threshold value t was set equal to the fraction of instances labeled as 1 (as opposed to 0). In red the OOB error rate for the 1s, in blue the 0s and in black the overall OOB error rate. On the y-axis is the OOB error rate, on the x-axis the number of trees. Results were obtained running a random forest with 500 trees for various choices of sentiment, matrix form and word count threshold. The n775-data set was used.

We see that the error rate for the instances whose true class is 1 can sometimes still be quite high. However, this is not too important too us. What we see in this picture is that, also with the lowered threshold, choosing $B = 500$ as the number of decision trees to grow in our random forest models seems like a safe choice. In all examples shown here, the error rate stabilizes long before this number of trees is reached. Thus we conclude that one method that solves the problem of class imbalance is running the random forest with a lowered threshold value for the predictions. We choose $B = 500$ as the number of trees to grow. Note that with regard to the computation of the AUC and the AUC_μ, we have not changed anything. The choice of a lower threshold value t (or different misclassification matrix A in the multinomial setting) only serves to give us confidence in our choice of B.

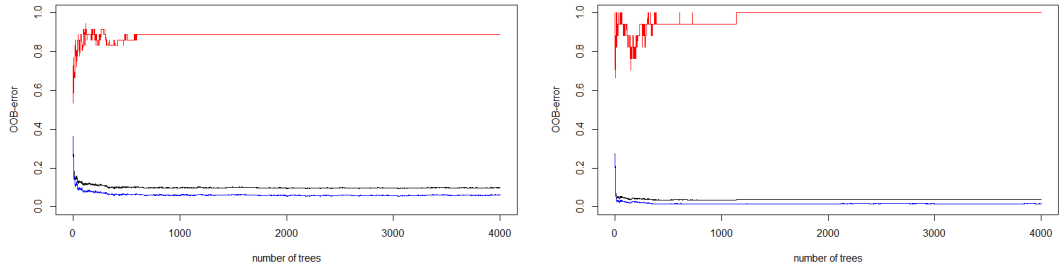
6.2.2. An alternative for dealing with class imbalance

Besides adapting the threshold value t (for the binary setting) or partition matrix A (for the multinomial setting), there is an additional method of dealing with the class skew. What causes a random forest to disproportionately allocate probability to for example class 0 (in the binary setting) is that decision trees are trained with bootstrap samples that contain a large number of instances of class 0. This, because these bootstrap samples are sampled randomly from the data set, where we have the same imbalance.

A different way to overcome this problem is to use stratified sampling. In this context, stratified sampling means that, given a data set with classes $\{1,\ldots,P\}$, we sample the same number of instances for every class $q \in \{1,\ldots,P\}$. Formally we write that when we make use of stratified sampling with sample size n_{strat}, we sample, for every class q, n_{strat} times from $\left\{\left\{\left(d'_{i,j}, \underline{x}_i\right)\right\} : d'_{i,j} = q, i \in \{1,\ldots,n\}\right\}$, with replacement. For the binary case, we sample n_{strat} times, with replacement, from $\left\{\left\{\left(d'_{i,j}, \underline{x}_i\right)\right\} : d'_{i,j} = 1, i \in \{1,\ldots,n\}\right\}$ and n_{strat}

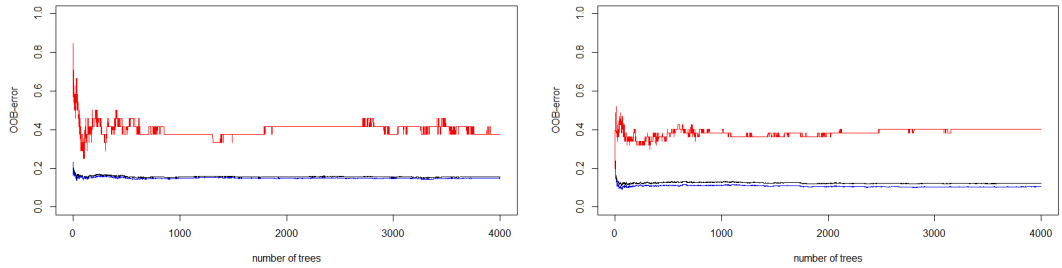
times from $\left\{\left\{\left(d'_{i,j}, \underline{x}_i\right)\right\} : d'_{i,j} = 0, i \in \{1,\ldots,n\}\right\}$.

Generally, it is safe to have n_{strat} be more or less equal to the the number of instances that are available for the least common class. So when q is the least common class in the data set, and n_q is the the size of the set $\left\{\left\{\left(d'_{i,j}, \underline{x}_i\right)\right\} : d'_{i,j} = q, i \in \{1,\ldots,n\}\right\}$, then we let $n_{\text{strat}} = n_q$. Take n_{strat} too large, and the decision trees will be more similar which causes the risk of overfitting. Take n_{strat} too small, and the decision trees will be extremely different, and we will then need a huge number of trees to still have an effective model. Since $n_{\text{strat}} < n$ a consequence of stratified sampling is that the trees are grown using a lower number of data points. Because we want our model to use all the data that we have, we choose to greatly increase the number of trees we grow. We set $B = 4000$.



(a) driver 13, positive sentiment, mean matrix form, word count threshold 21.

(b) driver 3, negative sentiment, normalized tf-idf matrix form, word count threshold 6.



(c) driver 8, negative sentiment, document term matrix form, word count threshold 16.

(d) driver 15, present sentiment, indicator matrix form, word count threshold 11.

Figure 6.7: Plot of the OOB error rate for a random forest in the binary setting, with a threshold value of $t = 0.5$. Stratified sampling was applied in sampling instances to grow the trees. In red the OOB error rate for the 1s, in blue the 0s and in black the overall OOB error rate. On the y-axis is the OOB error rate, on the x-axis the number of trees. Results were obtained running a random forest with 4000 trees for various choices for driver, sentiment, matrix form and word cont threshold in the n775-data set. The indicator matrix form was used as input.

We used the same combinations of driver, sentiment, matrix form and word count threshold as in Figure 6.6. We see that the OOB error rate for the instances with 1 for their true class stays at a higher level than the OOB error rate for the 0s. This does not necessarily concern us. We will judge the performance of the model later using the AUC. What we take away from Figure 6.7 is that $B = 4000$ is enough trees, because the OOB error rate stays stable.

Now we have seen two methods of implementing random forests for our problem. These are:

1. adapting the threshold t (in the binary setting) or partition matrix A (in the multinomial setting). In this way, we do not compute $\widehat{d}_j$ through majority vote, but set our own cut-off on the votes. We refer to this as the 'default option', because it is not a change to the default version of the random forest with regard to how we estimate p_j.

2. using stratified sampling for growing our decision trees. We drastically increase the number of trees grown and set $B = 4000$. We refer to this as the 'stratified option.'

6.3. Variable importance

We have established how the random forests work, and how we will use them in the context of this thesis. On top of that, we wish to gain some insight in the variables (words) deemed important by the random forest method for predicting the driver scores. In Chapter 5, we saw how the coefficients estimated as non-zero by the Elastic net models (with $\alpha \neq 0$) can be used, and we can shift the value of λ to get more/less non-zero coefficients. For the random forests, we have *variable importance* as a method to give us these, well, important variables.

In the previous section, we discussed the out of bag (OOB) instances and how we can use them to obtain the OOB error rate. The OOB instances are also used for the variable importance. When running the OOB instances down the decision trees, a number of correctly labeled instances is found. Let $x_v, v \in \{1,\dots,V\}$ now be the variable for which we wish to compute the variable importance. Only a limited number of the classification trees in the random forest are built using variable x_v. For the OOB cases of these trees, randomly permute the values of variable x_v. Then, run the OOB cases down the tree again. A different number of OOB instances labeled correctly is found. Take the difference of these two numbers, and then take the average of these differences among all trees that are built using variable x_v. We name this number the *raw importance score* for variable x_v.

Doing this repeatedly for all variables $x_1,\dots,x_V$, we find importance scores for all variables in our data set. We can then rank these and either set a threshold value or choose, for example, the 10 highest ranked variables. In this way, we can gain insight into which variables are important for the random forests when predicting the driver scores. Note though that this technique does not compensate for class imbalances, while it is influenced by it. AUC based variable importance measures have also been proposed, and shown to give better results for imbalanced data sets [3]. These are however not considered in this thesis.

7

Methodology

In this chapter, we will show how we apply the methods introduced in this thesis to obtain result. We will explain and motivate the choices made in this process and how we we can interpret these results.

7.1. Method

We have seen two types of methods in Chapters 5 and 6. In Chapter 5 we introduced the Elastic net models (including LASSO and Ridge regression), and in Chapter 6 we discussed the option of using Random Forests. Both these methods take annotated data as input to produce a predictive model for future reviews. In Chapter 3 we discussed the possible forms of document-term matrices as input for the Elastic net and Random Forest. In Chapter 4 we discussed how we can evaluate the performance of these models.

Now that we want to obtain our results, there are many choices to make. One has to choose which form to use for the variables, which model to use, how to evaluate the results et cetera. In this thesis the results are always obtained using a k-fold cross validation, regardless of which model is used. Note that this is a different cross validation than the one to determine λ in the Elastic net models. An example of a set of results could be:

driver number	AUC estimate
1	0.92
2	0.86
⋮	⋮
15	0.87

Figure 7.1: Results obtained for all 15 drivers using the n775-data set for present sentiment in the binary setting. Parameter were chosen as document-term matrix form, word count threshold set to 1 and a LASSO model (with a 5-fold cross validation to determine λ). Results were obtained using a 5-fold cross validation.

These are estimates for the AUC for all 15 drivers in the n775-data set. Say that we now change the matrix form, and compute the AUC estimates again. We get 2 sets of results.

driver number	AUC estimate
1	0.92
2	0.86
⋮	⋮
15	0.87

(a) document term matrix form.

driver number	AUC estimate
1	0.91
2	0.87
⋮	⋮
15	0.84

(b) mean matrix form.

Figure 7.2: Results obtained for all 15 drivers using the n775-dataset for present sentiment in the binary setting. Parameter were chosen as word count threshold set to 1 a LASSO model (with a 5-fold cross validation to determine λ) and various choices for the matrix form. Results were obtained using a 5-fold cross validation.

We change one part of our method and found different results. To see the influence of this change, we want to compare this two. But how do we do this? In this chapter, we will summarize the choices that have to be made in creating our model, and how to compare results obtained using one method with results obtained using another. Figure 7.3 gives an overview of how we obtain our results and illustrates all possibilities that we can vary.

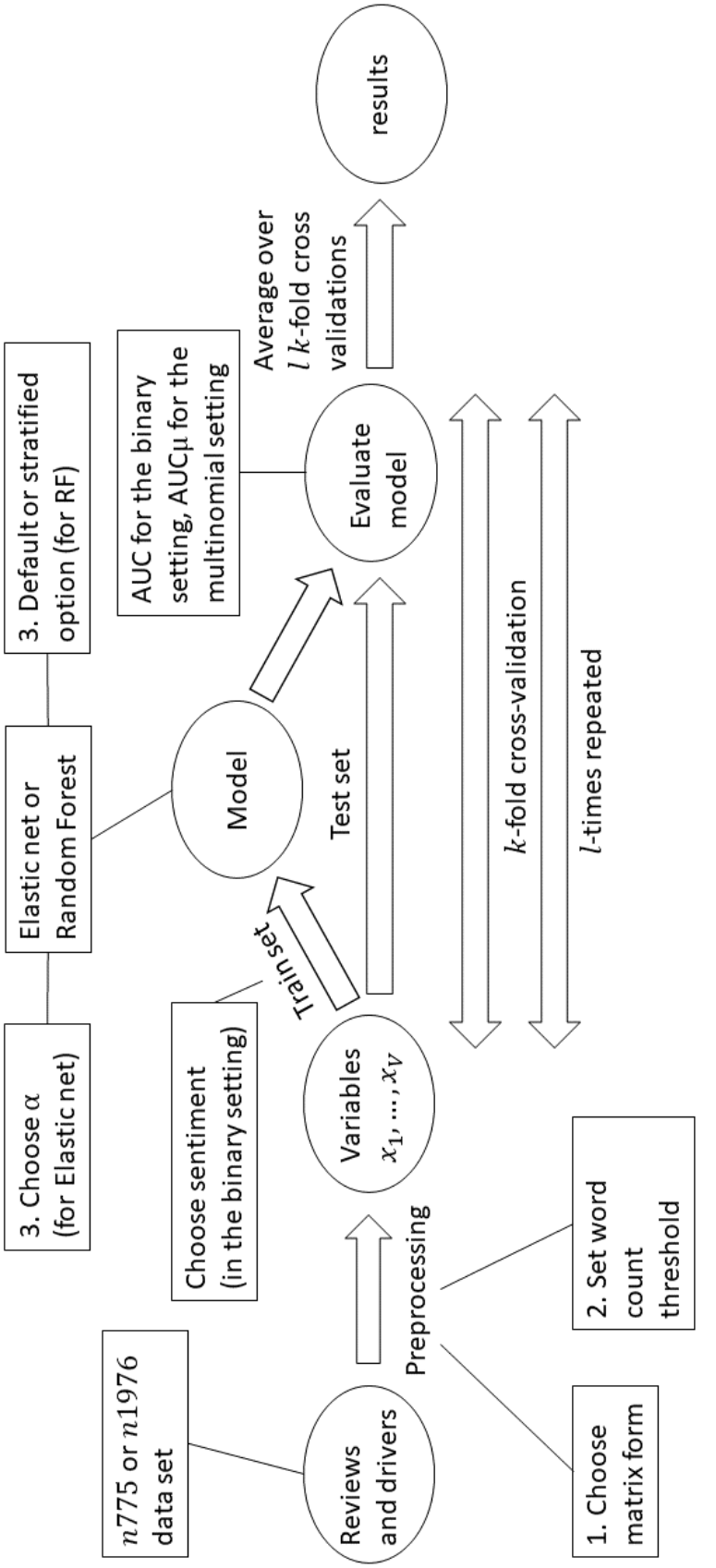


Figure 7.3: Illustrative overview of how we obtain results and what choices have to be made.

Figure 7.3 gives us an overview of the choices that have to be made when generating results. First of all, we have to determine what we want to predict. What data set are we going to use and, in the binary setting, what type of sentiment are we looking for? In the multinomial setting it is also important to set the misclassification matrix A. Which misclassifications do we consider costlier than others? We name these choices the *application choices*. They are determined by what one is looking to predict when applying the work of this thesis. From a modellers point of view, one cannot influence this. Given choices for these options, it is up to us to produce the best model. The application choices for these are shown in the table underneath.

		# of options	options
1.	data set	2	n1976-data set and n775-data set.
2.	sentiment	3	present, positive and negative (in the binary setting)
3.	Misclassification matrix	3	Only in the multinomial setting

Figure 7.4: All application choices that are to be set.

The different data sets and choices for sentiment have been discussed in Chapter 2. The different choices for the misclassification matrix will be discussed later. Within the data set, we run models for the individual drivers, but drivers has not been added as a factor in Figure 7.4. This, because when comparing models in this thesis, we will always look at average performance over all drivers in the data sets. We will look at the performance of our models for the individual drivers in the data sets later, but then it is out of interest in the differences between the drivers, not because we want to compare models.

Then, we move on the technical choices that we have to make. There are 3 that we will actively vary. We name these the *parameters* in our model. These have been numbered 1, 2 and 3 in Figure 7.3. The different choices for the parameters are summarized in the table underneath. The challenge is to find the 'optimal choice' for these parameters, i.e. the choice that gives us the best model.

	Parameter	# of options	options
1.	matrix form	5	standard, mean, indicator, tf-idf, normalized tf-idf
2.	word count threshold	5	$1, 6, 11, 16, 21$
3.	model	6	Elastic net for $\alpha \in [0, 0.5, 0.75, 1]$, two options for random forest

Figure 7.5: All parameters that we will actively vary with when obtaining our results.

The choices for the matrix form and word count threshold have been discussed in Chapter 3. The choice for model in Chapters 5 and 6.

Figure 7.3 gives the impression though that there are many more things that can be varied. And rightfully so, because there are a lot more options that we can play around with. We name these options the *redundant parameters*. Redundant, because we will show though that actively varying these options does not interest us. Either because varying them has no influence on our results, or because it is already clear to us varying them will have be useless, as we have already made our choice. These redundant parameters are given in Figure 7.6.

1. k in the k-fold cross validation
2. How to choose the λ for Elastic net models.
3. number of folds used in determining the λ for Elastic net models
4. Evaluation measure

Figure 7.6: The redundant parameters

The l-times repeated part of Figure 7.3 has not been discussed so far in this thesis. We will come to this later in this chapter, when discussing the variability of our results. In this chapter, we will show how we choose these redundant parameters and explain why we do not actively vary them. First, we will introduce the use of *effects plots* in this chapter and, discuss the variability of our results and then introduce the option of using

l times repeated computations. Then, we will show how to analyze the variance of our results. Lastly, using some of this theory, we will show why we do not actively vary the redundant parameters listed in Figure 7.6.

7.2. Effects plots and analysis of variance

What is interesting to us is the effect the use of the parameters, and up and till now also the use of the redundant parameters, has on the performance of our model. Which combination of parameter settings gives the most predictive power? And how do we effectively compare them? Given the number of options, it will be a lot of manual work to compare the results of every possible combination manually.

Take the example introduced in Figure 7.2 and combine the two tables into one.

matrix form (→), driver number(↓)	document term	mean
1	0.92	0.91
2	0.86	0.87
⋮	⋮	⋮
15	0.87	0.84

Figure 7.7: Results obtained for all 15 drivers using the n775-dataset for present sentiment in the binary setting. Parameter were chosen as word count threshold set to 1 and a LASSO model (with a 5-fold cross validation to determine λ) and 2 different choices for the matrix form, document term and mean matrix form. Results were obtained using a 5-fold cross validation.

So, how will we compare these results? For some drivers the result has slightly improved, for some it has decreased and for others it stayed the same. Figure 7.7 gives the impression that there are 2 factors influencing these AUC outcomes: the matrix form and the driver number. There is another matter of influence though: randomness. The results are always obtained using a k-fold cross validation. Furthermore, when the chosen model is an Elastic net model, as is the case in Figure 7.7, a different k-fold cross validation is used to determine the λ parameter for the model. The folds in the cross validations are sampled at random, and this is where the randomness comes in. This makes that when computing the AUC estimate for one of the drivers as in Figure 7.7, the probability of obtaining the exact same result twice is close to zero, also when the exact same model was used. We thus not only have the question of how to compare the results, but also how to deal with the randomness in the results.

A solution to the first part of the question is found in the average values. The mean of all AUC values in Figure 7.7 is 0.815. We expect deviations from this mean among the results to be caused by the driver number and/or matrix form. Meanwhile, we have to think of the randomness in our results, as was discussed earlier. This is the foundation of the Analysis of Variance (ANOVA) model, which we will use to interpret our results.

In the ANOVA model, we assume that we have a set of outcomes, such as the AUC values in Figure 7.2. Let these outcomes be y_i, $i \in \{1,\dots,n\}$ and let $\overline{y}$ be the mean of these samples, i.e. $\overline{y} := \frac{1}{n}\sum_{i=1}^{n} y_i$. We assume that y_i are outcomes of random variables Y_i, whose distribution has mean μ. We assume that the variance of y_i around μ is caused by a number of known factors and a random error. Let these factors be variables $X_1,\dots,X_m$, $m \in \mathbb{N}$, and let $x_{i,j}$ be the outcome of variable X_j for outcome y_i. In Figure 7.2 we for example had the factors matrix form, X_1, and driver number, X_2, and had $x_{1,1}$ = document term and $x_{1,2} = 1$. The assumption now made in the ANOVA model is that, given the outcomes of the factors, Y_i, or now $Y_i|x_{i,1},x_{i,2}$, is distributed as

$$Y_i \sim \mathcal{N}\left(\mu + \beta_1\left(x_{i,1}\right) + \beta_2\left(x_{i,2}\right), \sigma^2\right) \tag{7.1}$$

where $\sigma^2 > 0$ is the variance. Alternatively, we can write

$$Y_i = \mu + \beta_1\left(x_{i,1}\right) + \beta_2\left(x_{i,2}\right) + \epsilon_i \tag{7.2}$$

where $\epsilon_i \sim N\left(0,\sigma^2\right)$ i.i.d. for all i. This is the ANOVA model with only the *main effects* of the factors X_1 and X_2. It is assumed that the main effects are additive. The main effects in Equation 7.1 can be estimated using our data. For example, we find that the mean value of the AUC values found when using the document term

matrix is 0.821 and when using the mean matrix form it is 0.808. Thus, the estimate of the main effect of the document term matrix form is $\widehat{\beta}_1\left(\text{document term}\right) = 0.821 - 0.815 = 0.6$ and of the mean matrix form is $\widehat{\beta}_1\left(\text{mean}\right) = 0.808 - 0.815 = -0.7$ (these differences should be each other's opposites, the difference is due to rounding errors). The same can be done for the main effects of the driver numbers and this gives us a model that, given $x_{i,1}, x_{i,2}$ can predict y_i as

$$\widehat{y}_i = \overline{y} + \widehat{\beta}_1\left(x_{i,1}\right) + \widehat{\beta}_2\left(x_{i,2}\right) \tag{7.3}$$

. When using the ANOVA model, three assumptions are made about the probability distribution of the responses. These are

1. The observations are independent.
2. The distributions of the residuals are normal.
3. The variance of the data within groups should be the same.

Let us briefly discuss if these assumptions apply. In our example and in the rest of this thesis, all responses are independent, so the first assumption is met.

The second assumption poses a slight problem to us. The residuals are $y_i - \widehat{y}_i$. Use of the ANOVA assumes that the distributions of the residuals are normal. This is tricky. The AUC scale of is not totally linear. First, the AUC values cannot be larger than 1. Imagine if in Equation 7.1 $\mu + \beta_1\left(x_{i,1}\right) + \beta_2\left(x_{i,2}\right)$ would be extremely close to 1. If the residuals are normally distributed, the outcome of the error term could make that a draw from the distribution might be higher than 1. Our results, here and in the rest of this thesis, are never that close to 1 though. Furthermore, we do not expect that the distribution of the residuals is centred around the mean. There will be more outliers below than above the mean of the residuals, and the ones below the mean will also be distanced further from the mean than those above. This because on the AUC scale it is 'harder' to increase the performance of a model from for example 0.7 to 0.8 than it is to increase it form 0.6 to 0.7. Therefore we expect that the residuals of the AUC estimates of the models will not evenly be distributed around the mean. For now, we accept that the assumption that the distributions of the residuals are normal is not entirely justified.

Then the third and last point. The assumption is that the variance among all groups is the same. A group is all observations that have the same value for one or multiple factors. However, we expect that the variance among can be different for for example different drivers and different types of sentiment. We will later address this in further detail. For now, we accept this and choose to apply the ANOVA model.

7.2.1. Sums of squares

This model is not perfect, and we are interested in the difference between our model and the observations. Decomposing the difference between the our observations y_i and the mean $\overline{y}$ can give some insight in this. We decompose into an explained part, meaning explained by our model, and a residual part, build up of variance from the random error and *model lack of fit*. This last term will be explained later. For the decomposition, we look at $\sum_{i=1}^{n}\left(y_i - \overline{y}\right)^2$, the so called *sum of squares*. Let $\widehat{\epsilon}_i$ be $y_i - \widehat{y}_i$, then

$$\sum_{i=1}^{n}\left(y_i - \overline{y}\right)^2 = \sum_{i=1}^{n}\left(\widehat{y}_i - \overline{y} + y_i - \widehat{y}_i\right)^2 \tag{7.4}$$

$$= \sum_{i=1}^{n}\left(\left(\widehat{y}_i - \overline{y}\right)^2 + 2\widehat{\epsilon}_i\left(\widehat{y}_i - \overline{y}\right) + \left(\widehat{\epsilon}_i\right)^2\right) \tag{7.5}$$

$$= \sum_{i=1}^{n}\left(\widehat{y}_i - \overline{y}\right)^2 + \sum_{i=1}^{n}\left(\widehat{\epsilon}_i\right)^2 + 2\sum_{i=1}^{n}\widehat{\epsilon}_i\left(\widehat{y}_i - \overline{y}\right) \tag{7.6}$$

$$= \sum_{i=1}^{n}\left(\widehat{y}_i - \overline{y}\right)^2 + \sum_{i=1}^{n}\left(\widehat{\epsilon}_i\right)^2 + 2\sum_{i=1}^{n}\widehat{\epsilon}_i\left(\overline{y} + \widehat{\beta}_1\left(x_{i,1}\right) + \widehat{\beta}_2\left(x_{i,2}\right) - \overline{y}\right) \tag{7.7}$$

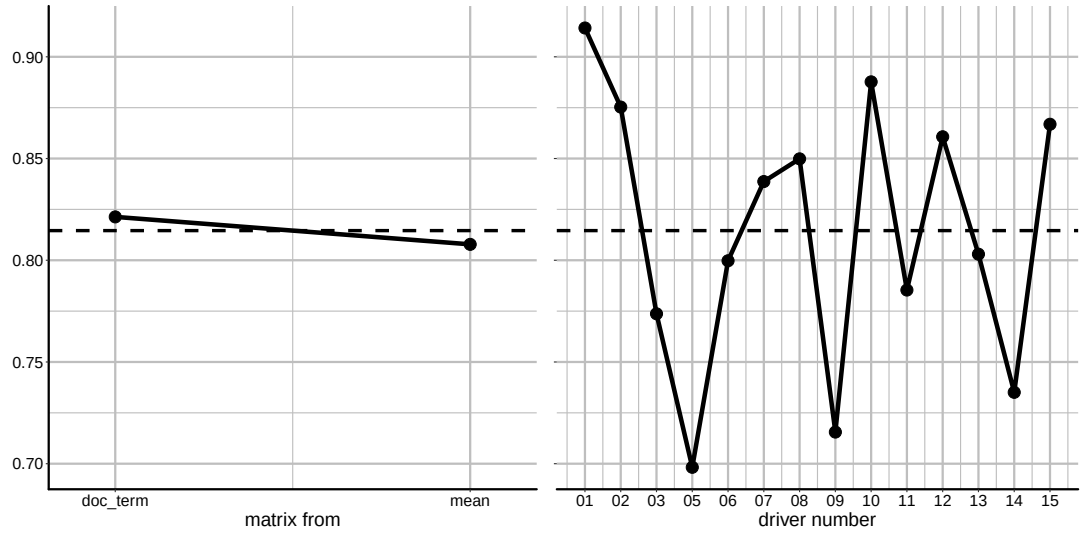
$$= \sum_{i=1}^{n}\left(\widehat{y}_i - \overline{y}\right)^2 + \sum_{i=1}^{n}\left(\widehat{\epsilon}_i\right)^2 + 2\sum_{i=1}^{n}\widehat{\epsilon}_i\widehat{\beta}_1\left(x_{i,1}\right) + 2\sum_{i=1}^{n}\widehat{\epsilon}_i\widehat{\beta}_2\left(x_{i,2}\right) \tag{7.8}$$

$$= \sum_{i=1}^{n}\left(\widehat{y}_i - \overline{y}\right)^2 + \sum_{i=1}^{n}\left(\widehat{\epsilon}_i\right)^2 \tag{7.9}$$

The last step follows, because by design we have that $\sum_{i=1}^{n} \widehat{\epsilon}_i = \sum_{i=1}^{n} y_i - \widehat{y}_i = 0$[10]. In short, this is because the estimates $\widehat{\beta}_1$ and $\widehat{\beta}_2$ are chosen as means for the factors over the data y_i. So when we sum over $i = 1$ to $i = n$ the difference between $\widehat{y}_i = \overline{y} + \widehat{\beta}_1 \left(x_{i,1}\right) + \widehat{\beta}_2 \left(x_{i,2}\right)$, the result is 0.

This was done for an ANOVA model with 2 factors of which only the main effects were incorporated, as in the example from Figure 7.7. It holds for any m number of factors though, including any *interaction* or higher order effects. We introduce these effects later.. $\sum_{i=1}^{n} \left(\widehat{y}_i - \overline{y}\right)^2$ is called the *explained sum of squares*, abbreviated as SS_{expl}, and $\sum_{i=1}^{n} \left(\widehat{\epsilon}_i\right)^2$ the *residual sum of squares*, abbreviated as SS_{res}. To distinguish it from the other two, $\sum_{i=1}^{n} \left(y_i - \overline{y}\right)^2$ is also called the *total sum of squares*, abbreviated as SS_{total}. We thus have that $\text{SS}_{\text{total}} = \text{SS}_{\text{expl}} + \text{SS}_{\text{res}}$.

Furthermore, we introduce the *group sum of squares*, abbreviated as SS_{group}. As an example, take the factor X_1 from our example, matrix form. There are two groups within this factor, 'document term' and 'mean'. Let n_1, n_2 be the number of observations y_i obtained with a document term or mean matrix form respectively(these numbers are the same in our example). Let $\overline{y}_{1,1}$ be the mean of all observations obtained using a document term matrix form, $\overline{y}_{1,2}$ be the mean of all observations obtained using a mean matrix form. Then the group of sum of squares for the matrix form, $\text{SS}_{\text{group}}\left(X_1\right)$, is calculated as $\sum_{i=1}^{2} n_i \left(\overline{y}_{1,i} - \overline{y}\right)^2$. In the ANOVA model, we often work with the means of groups. Plotting the means for all groups for our 2 factors, gives us the plots in Figure 7.8.



(a) Effect of matrix form.

(b) Effect of driver number.

Figure 7.8: Example of the main effects plots for matrix form and driver number, using the results from Figure 7.7. The dashed line gives the mean of all observations.

We name these plots the *main effects plots*. Judging from this picture, it seems that the effect of the driver number is larger than the effect of the mean. A different choice of driver number can lead to a much larger deviation from the overall mean than a different choice of matrix form. It is clear that $\text{SS}_{\text{group}}\left(X_2\right)$, the group sum of squares for the driver number factor, wil be larger than $\text{SS}_{\text{group}}\left(X_1\right)$. Do not be too quick to draw any other conclusions from this observation though. We have not accounted for the variance, or SS_{res} in general yet.

Earlier, we showed that $\text{SS}_{\text{total}} = \text{SS}_{\text{expl}} + \text{SS}_{\text{res}}$. If we were to remove a factor, say matrix form, from our model (so that in our example, we would model y_i as being only influenced by driver number and noise), SS_{expl} would decrease and SS_{res} would increase, so that SS_{total} stays the same. The number by which SS_{expl} decreases equals $\text{SS}_{\text{group}}\left(X_1\right)$ for the factor of matrix form. The factor is no longer in our model, and the variance in our results is no longer explained by it. SS_{res} is thus not only build up by variance of the noise, but also by the *model lack of fit*, factors not included in our model.

In our example, we only varied with matrix form and driver number. There lack of fit is not caused by the influence of another parameter not discussed so far, but there is another factor we can introduce. This factor is the interaction effect between the matrix form and driver number, denoted as $\widehat{\beta}_{1,2}$. This is what is named two-way ANOVA. The assumption from the one-way ANOVA from Equation 7.1 is changed to include the interaction effect.

$$Y_i \sim \mathcal{N}\left(\mu + \beta_1\left(x_{i,1}\right) + \beta_2\left(x_{i,2}\right) + \beta_{1,2}\left(x_{i,1}, x_{i,2}\right), \sigma^2\right) \tag{7.10}$$

We can also compute the sum of squares for the interaction effect. Let a and b be the number of groups formed by X_1 and X_2 respectively and let $\overline{y}_{i,j}$ be the average of the $n_{i,j}$ observations where $X_1 = i$ and $X_2 = j$, $i \in \{1,\ldots,a\}, j \in \{1,\ldots,b\}$. Note that in our example $n_{i,j} = 1\ \forall i,j$, meaning that $\overline{y}_{i,j}$ is the 'average' of 1 observation. We will be in the situation where $n_{i,j} > 1$ when we come to the results in Chapter 8. Furthermore, let $\overline{y}_{i,\cdot}$ be the overage over all observations where $X_1 = i$ and $\overline{y}_{\cdot,j}$ the average over all observations where $X_2 = j$. The sum of squares for the interaction effect is then computed as

$$\sum_{i=1}^{a}\sum_{j=1}^{b}\left(\overline{y}_{i,j} - \overline{y}_{i,\cdot} - \overline{y}_{\cdot,j} + \overline{y}\right)^2 \tag{7.11}$$

This idea can be expanded to having m main effects; there are then $m(m-1)/2$ interaction effects. We can also plot these interaction effects. We name these figures the *interaction effects plots*. For our example, we get:

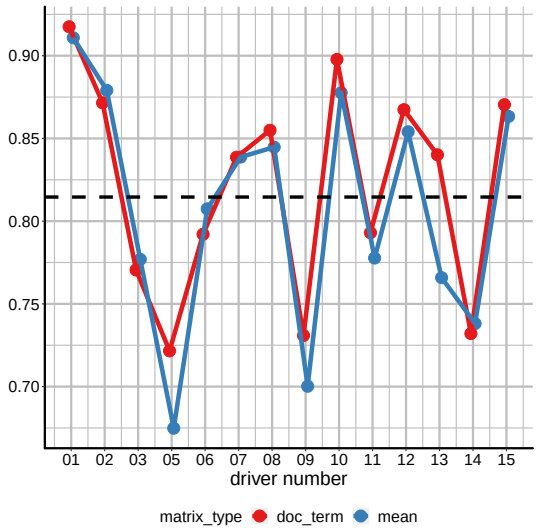


Figure 7.9: Example of the interaction effects plot for the interaction between matrix form and driver number, using the results from Figure 7.7. The dashed line gives the mean of all observations.

Note that any point in this graph is simply one of the results from Figure 7.7. It is thus no longer an average of multiple results. We say that we have now fully specified our model. Any variation in the outcomes not explained by the main effects of the matrix form and driver number, is explained by the interaction effect. This means that we now have $y_i = \widehat{y}_i$, thus that SS_{res} $(= \sum_{i=1}^{n}\left(y_i - \widehat{y}_i\right)^2)$ equals 0, which means that $\text{SS}_{\text{total}} = \text{SS}_{\text{expl}}$. This means that this model is not too good of a model, because, as mentioned earlier, there is some variance in each result from Figure 7.7, due to randomly sampling of the folds. To illustrate this effect, take a look at Figure 7.10.

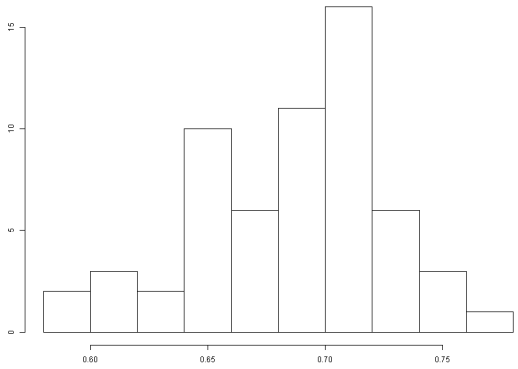


Figure 7.10: Histogram of 60 results obtained in the binary setting using the $n775$-data set. Result is the AUC value found using a LASSO with 5-fold cross validation to find λ. The aim was prediction of positive sentiment for driver 1, 'Charging Electronic Power'. The other parameters used were document-term matrix form for the variables, using a word count threshold of 1.

What we see is a histogram of 60 results for a specific choice of parameters. We see that the results range from an AUC of lower than 0.60 to higher than 0.75. A higher number of folds could help, but we run into a different problem then. Having a higher number of folds in our cross validations could negate this effect, but then a different problem comes up. We must have a sufficient number of instances from every class in our test set. A large increase in the number of folds will lead to very few 1s in the test set, or worst case none. An AUC estimate based on a test set without 1s is not possible, and with only one, will be extremely volatile, and for some drivers we have quite a limited number of reviews annotated as 1. A possible way to overcome this problem would be to repeat our entire computation $l > 1$ times, and then take the average of the outcomes. This comes at a computational cost though. Therefore we will often simply run $l = 1$ time for the analysis of our results, but we will have to be cautious about drawing any conclusions, because we know of the variability.

In the example from Figure 7.7 that we have used so far, we only varied driver number and matrix form and created a two-way ANOVA model based on these two factors and the interaction factor. In the results as discussed in Chapter 8, we will also vary the sentiment, word count threshold and model, and create two-way ANOVA model using these factors and their interaction factors. It is then also possible to model 3rd order interaction, or even higher order interactions, but we will not do that within this thesis. Note that with the introduction, the two-way ANOVA model will no longer be fully specified, as interaction effects of order higher than 2 are not taken into account. Every point in an interaction plot as in Figure 7.9 will be the average of all outcomes given by the unique combination of the two factors depicted in the plot. This will give us some more confidence in the effects shown in these plots. Up to us to determine whether the difference in effects is significant, i.e. that one point being higher than another is the result of the change in factor values, or insignificant, meaning that the variability in these points makes that we cannot draw any conclusions.

7.2.2. Variance and significance

To find whether the effect of a factor is significant or not, we will conduct an *F-test*. To explain how, we do this, we will first need to introduce the concepts of *degrees of freedom* and *mean square*. We will also use these to estimate the variance of Y_i.

When we look at a factor, say X_1 we say that the degrees of freedom of this factor, $\mathrm{DF}(X_1)$ equals the number of groups in this factor minus 1. In the example of Figure 7.7 we find that the factor matrix form has $2-1=1$ degree of freedom and the factor driver number has $15-1=14$ degrees of freedom. The degrees of freedom of an interaction factor is found by multiplying the degrees of freedom of the corresponding main effects. In case of our example, the degrees of freedom for the interaction between X_1 and X_2 thus becomes 14. We also associate a value to the degrees of freedom of the entire model. We name this $\mathrm{DF}_{\mathrm{expl}}$. This is found by the summing the degrees of freedom of all factors, both for the main effects as well as for the interaction effects. For example, the fully specified model as in Equation 7.10 has $1+13+13=27$ degrees of freedom. Lastly, the degrees of freedom for the residuals is the number of degrees the model would have when it is fully specified, minus the number of degrees it actually has. The residual in the one-way ANOVA model as in Equation 7.1 for example has $27-13-1=13$ degrees of freedom (= the degrees of freedom the factor for the interaction

between X_1 and X_2 has). We name the residual degrees of freedom $\mathrm{DF}_{\mathrm{res}}$.

The mean square, MS of a factor is found by dividing the sum of squares of a factor by its degrees of freedom. Thus, for a factor X we would then have,

$$\mathrm{MS}_{\mathrm{group}}(X) = \frac{\mathrm{SS}_{\mathrm{group}}(X)}{\mathrm{DF}(X)} \tag{7.12}$$

We not only introduce the mean square for the factors and their interactions, but also for the whole model and the residuals. We then have

$$\mathrm{MS}_{\mathrm{expl}} = \frac{\mathrm{SS}_{\mathrm{expl}}}{\mathrm{DF}_{\mathrm{expl}}} \tag{7.13}$$

$$\mathrm{MS}_{\mathrm{res}} = \frac{\mathrm{SS}_{\mathrm{res}}}{\mathrm{DF}_{\mathrm{res}}} \tag{7.14}$$

$\mathrm{MS}_{\mathrm{res}}$ is then an estimate for σ^2 [10], the variance introduced in Equation 7.1. Since this variance is assumed to be the same among all Y_i, we have that the variance within each group is the same. $\mathrm{MS}_{\mathrm{res}}$ can thus be regarded as measuring the variation within groups, whereas $\mathrm{MS}_{\mathrm{group}}(X)$ measures variation among the groups. In Chapter 8, we will be looking at the means of groups. Since the variation is equal in each group, the standard error of the mean of a group, can then be computed as $\sqrt{\frac{\mathrm{MS}_{\mathrm{res}}}{n_{\mathrm{group}}}}$, n_{group} being the number of observations within the group.

Now we introduce the F-test. For a factor X, the F-statistic is computed as

$$F = \frac{\mathrm{MS}_{\mathrm{group}}(X)}{\mathrm{MS}_{\mathrm{res}}} \tag{7.15}$$

The null-hypothesis is that there the populations representing the groups of factor X are the same, i.e. the outcomes from both groups are draws from the same distribution. In case that there is no difference, we have that $\mathbb{E}\left(\mathrm{MS}_{\mathrm{group}}(X)\right) = \mathbb{E}\left(\mathrm{MS}_{\mathrm{res}}(X)\right)$, and thus that the F-statistic will be about 1. Under the null-hypothesis, the F-statistic then has an F-distribution. The F-distribution is specified by two degrees of freedom. In our case, these are $\mathrm{DF}(X)$ and $\mathrm{DF}_{\mathrm{res}}$. When we have computed the F-statistic, we can compute the p-value, the probability of observing the value found or a greater value when we think of it is a draw from the F-distribution with $\mathrm{DF}(X)$ and $\mathrm{DF}_{\mathrm{res}}$ as the degrees of freedom. We need to set a significance level α, such that we reject the null-hypothesis when $t < \alpha$. Within this thesis, we will generally choose $\alpha = 0.05$. Rejection of the null-hypothesis then means that the groups (or, in case of an interaction effect, unique combinations of groups) are not from the same population. We then say that the effect of the factor is significant. Performing the F-test for all factors gives what is known as the ANOVA table. For the one-way ANOVA model of our example, we construct the following ANOVA table:

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
matrix_type	1	0.00	0.00	4.44	0.0552
driver_number	13	0.12	0.01	31.13	0.0000
Residuals	13	0.00	0.00		

The columns represent, in order from left to right, the degrees of freedom, $\mathrm{SS}_{\mathrm{group}}$ or $\mathrm{SS}_{\mathrm{res}}$, $\mathrm{MS}_{\mathrm{group}}$ or $\mathrm{MS}_{\mathrm{res}}$, F-statistic, p-value. The first two rows give the statistics for the factors in the one-way ANOVA model, the last row for the residuals.

7.2.3. Generating results

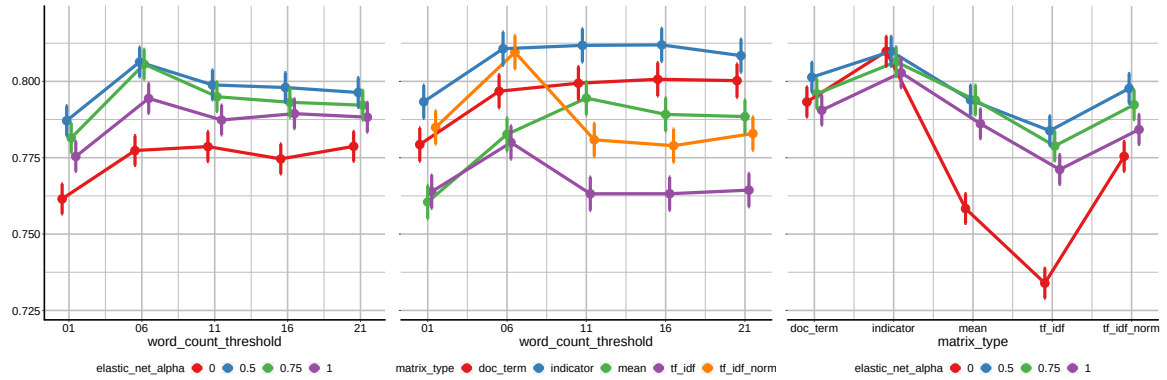
In the example from Figure 7.7 we had 2 factors, driver number and matrix form. When we will generate results in Chapter 8, we will have 3 more factors: sentiment, word count threshold and model. The matrix form, word count threshold and model are the parameters, as described earlier in this chapter in Figure 7.5. The Figure is reprinted here:

	Parameter	# of options	options
1.	matrix form	5	standard, mean, indicator, tf-idf, normalized tf-idf
2.	word count threshold	5	$1, 6, 11, 16, 21$
3.	model	6	Elastic net for $\alpha \in [0, 0.5, 0.75, 1]$, two options for random forest

Figure 7.11: All parameters that we will actively vary with when obtaining our results.

For sentiment, we have present, positive and negative sentiment. For the driver numbers, we always generate AUC values for all drivers in the data set used, unless this is not possible. When it is not possible, this due to a lack of data for that driver, there need to be enough 1s (or, for the multinomial setting, Ps and Ns) to be able to compute an AUC estimate without the variance of this estimate being too volatile. We say that the number of 1s needed (or, for the multinomial setting both Ps and Ns) is twice the number of folds used in the cross validation. What this means that in the binary setting there are $3 \times 5 \times 5 \times 6 = 450$ unique combinations for these 3 parameters and sentiment, and we thus obtain 450 columns of results like the 2 we had in Figure 7.7. This also further clarifies why repeating computations (choosing $l > 1$) and taking the average becomes computationally too expensive.

We will then plot the results of the effects of the parameters. This, because we are interested in creating the model that on average performs best for all choices of sentiment and driver number. We could theoretically create separate models, for every type of sentiment, driver number, or even every unique combination of the two, but this would require $l > 1$ computations again, and would furthermore be a lot of manual analyzing for this thesis. Thus, we create a two-way ANOVA model using these 5 factors and plot the main effects and interaction effects plots for the parameters. An example of these interaction plots is given in Figure 7.12.



(a) Interaction effect of word count threshold vs. Elastic net model.

(b) Interaction effect of word count threshold vs matrix form.

(c) Interaction effect of matrix form vs Elastic net model.

Figure 7.12: Interaction effects plot for the effect of the matrix form, word count threshold and Elastic net α. On the y-axis is the AUC value, on the x-axis the first varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times$ SE. Results were obtained using the n1976-data set running the models for all sentiments. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

These figures were obtained by applying the discussed procedure, using the n1976-data set and only using Elastic net models, no random forests. These Figures also show how we can easily gain insight in these large numbers of results. From these plots, and those of the main effects, we can easily deduce which models are performing better and which worse. For example, it is easy to see that wen looking at the interaction of word count threshold and model, the best performance seems to be achieved by a word count threshold of 6 and the elastic net models with $\alpha = 0.5$ or $\alpha = 0.75$. Beware of the uncertainty of the results though. We have plotted confidence intervals, calculated as $1.96\times$ standard error. These standard errors are computed using MS_{res} as an estimate for the variance, as explained in the previous section. We can also compute the earlier introduced ANOVA table for all factors for the main and interaction effects.

We see that in this case all factors and their interactions are deemed significant, except the interaction between the Elastic net model and the word count threshold. This means that the choice of Elastic net model and word count threshold certainly have an effect on the outcome, but the choice of Elastic net model has no

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
matrix_type	4	0.80	0.20	143.99	0.0000
driver_number	14	20.40	1.46	1046.09	0.0000
elastic_net_alpha	3	0.35	0.12	83.32	0.0000
sentiment	2	3.50	1.75	1255.86	0.0000
word_count_threshold	4	0.18	0.05	32.79	0.0000
matrix_type:driver_number	56	0.33	0.01	4.19	0.0000
matrix_type:elastic_net_alpha	12	0.27	0.02	16.45	0.0000
matrix_type:sentiment	8	0.14	0.02	12.71	0.0000
matrix_type:word_count_threshold	16	0.20	0.01	9.01	0.0000
driver_number:elastic_net_alpha	42	0.86	0.02	14.64	0.0000
driver_number:sentiment	28	8.19	0.29	209.92	0.0000
driver_number:word_count_threshold	56	0.14	0.00	1.84	0.0002
elastic_net_alpha:sentiment	6	1.13	0.19	135.07	0.0000
elastic_net_alpha:word_count_threshold	12	0.02	0.00	1.08	0.3742
sentiment:word_count_threshold	8	0.07	0.01	6.66	0.0000
Residuals	4228	5.89	0.00		

influence on the effect of the word count threshold and vice versa. When discussing the results in Chapter 8, we will not continuously print the ANOVA table, but only mention the interesting observations we make from these tables. First, we will discuss the redundant parameters as given in Figure 7.6, and show why we consider them redundant, sometimes with the help of some effects plots.

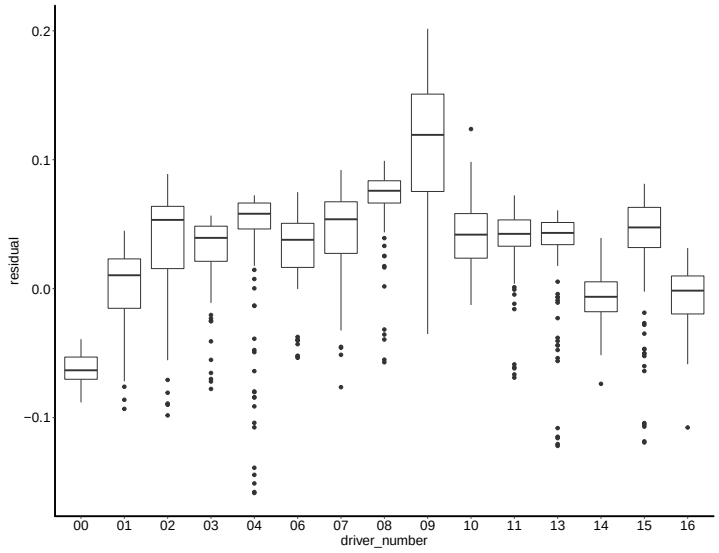
7.2.4. Criticism

Earlier in this chapter, we stated that the assumptions of the ANOVA model were not fully met. We stated that the residuals are probably not normally distributed and the suspected that the variance of the data within groups is not the same for all groups. We suspect that this variance may be dependent on the driver number and sentiment. The assumption made in the ANOVA model that the observations are drawn from distributions that all have the same variance σ^2 can then be questioned. We make boxplots for the residuals for all unique combinations of driver and sentiment from the ANOVA model built earlier in this section. The results are shown in Figure 7.13.

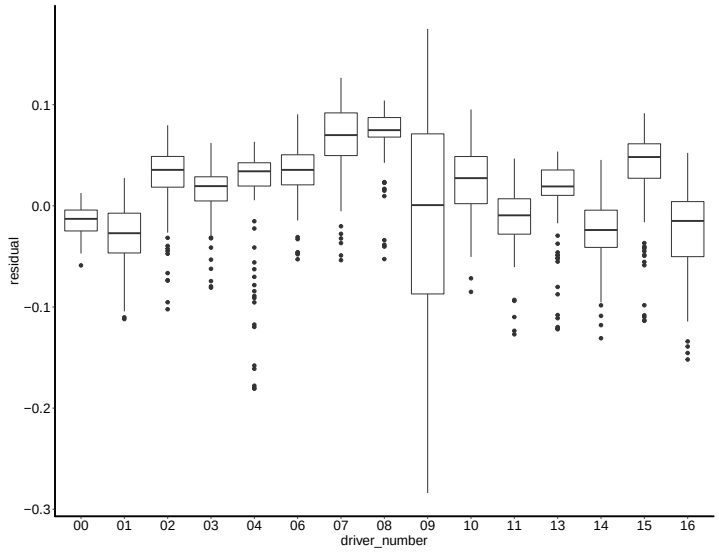
Regarding the distribution of the residuals, we do, as expected, see more negative outliers than positive ones. Fortunately these seem to be outliers. the distributions themselves do not seem to be too far tilted to undermine the ANOVA model.

Then we have the variance assumption. The variance of the residuals is most certainly affected by the choice of both driver number and sentiment. For example, we see that there is generally more variance in the residuals for driver number 9 and the variance in the residuals seems larger for negative sentiment than for positive or present sentiment.

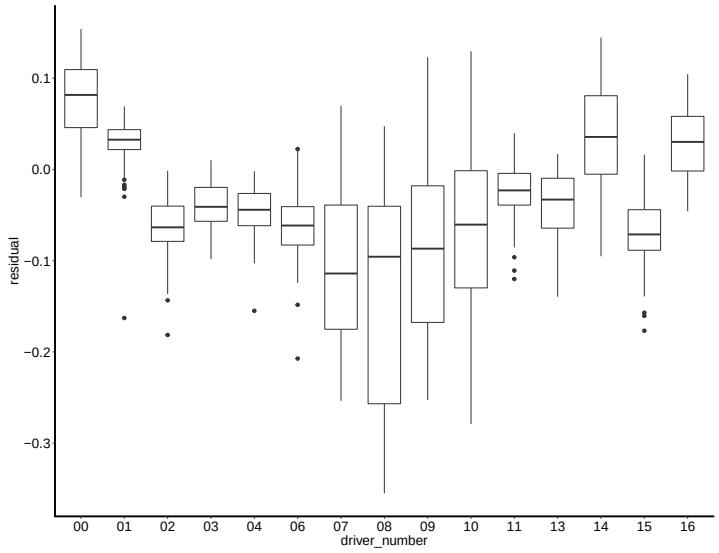
Despite these concerns, we still apply the ANOVA model and we will use effect plots as the one in Figure 7.12 to compare the effects of the parameters. The results from the boxplots in Figure 7.13 warn us though that the assumption of equal variance σ^2, and thus by direct implication the standard error and the plotted confidence intervals, has to be taken with a grain of salt.



(a) Present sentiment



(b) Positive sentiment



(c) Negative sentiment

Figure 7.13: Boxplots of the residuals of the observations compared to the group mean for all unique combinatons of sentiment and driver.

7.3. Redundant parameters

7.3.1. Choice of k, number of folds.

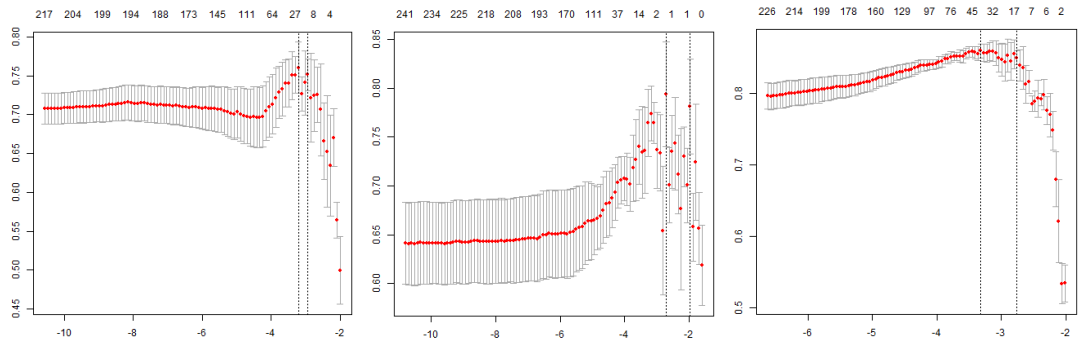
We validate our results using a cross validation. Note that this is a different cross validation from the one used in the Elastic net models for determining the value of λ. We can run a cross validation, within which our model is trained using a train set, evaluate its performance on the test set, and average the results for the multiple folds in the cross validation. This cross validation could be done for both 5 and 10 folds. A problem occurs though. In the binary setting we use the AUC as the evaluation measure and in the multinomial setting we use the AUC_μ. These are computed as in Equations 4.42 and 4.51. To compute these measures, there need to be sufficient data of each class in the test fold. In the binary setting, we cannot compute the AUC when there are only 0s in the test fold, and in the multinomial setting, we cannot compute the AUC_μ when N or P are not present as classes in the test fold. Thus, we need to sample the test sets and train sets in the cross validation in a stratified manner. We make them such that the proportion of classes within the sets is the same as in the data set.

Now back to actually choosing k. We choose to take $k = 5$ when obtaining our results. We say that we want at least 2 instances from the minority class in the test fold. We will see in Chapter 8 that this is not possible for all drivers in the data sets, due to a lack of instances from the minority class. Taking a higher number for k would therefore lead to a problem with the stratified sampling of the test and train set. When taking $k = 10$ in the binary setting for example, we would sample 0.1 from the total number of 1's in the data set for the test set. For some driver numbers in the data set, this leads to so few 1's that the AUC estimate based on the test set becomes impossible to make, because of a shortage of 1's. That is why we keep $k = 5$.

7.3.2. Choosing λ for Elastic net models

In both the binary and multinomial setting, Elastic net models use a cross validation to determine the λ parameter. The number of folds used in this process has to be set. Furthermore, as discussed in chapter 5, there is some discussion to be had on which λ from a series of 100 to choose after the cross validation. In the results presented in this thesis, we choose the λ with the highest AUC (in the binary setting) / AUC_μ (in the multinomial setting) value. This, as opposed to choosing a model with a higher value for λ that is thus more regularized. We will illustrate this choice. Let us print series of λ's for a number of different models. The results are shown in Figure 7.14.

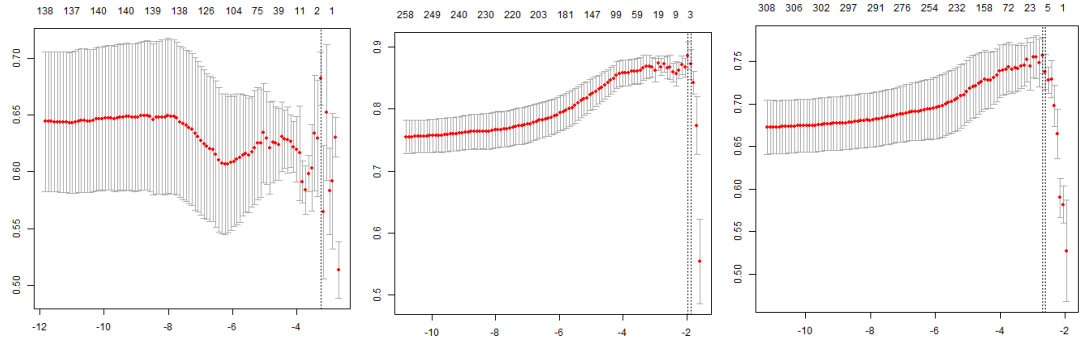
Note that the number of non-zero coefficients stays the same for the Elastic net models where $\alpha = 0$. This corresponds to Ridge regression, meaning that either all coefficients are non-zero or non of them. Now take a look at the models given by values of λ close the the value that gives the model with the maximum AUC estimate. The estimated AUC is lower, of course, but close the the highest estimated AUC we find. A model with a value for λ that is more regularized and whose estimated performance is still within one standard error from the maximum AUC model, can not always be found. In Figure 7.15 we have done the same thing for the multinomial setting. Here, we measured AUC_μ. The same observation as in the binary setting holds. Therefore, choosing the λ that gives the highest AUC or AUC_μ estimate becomes our standard procedure for when we use Elastic net models.



(a) driver 6, present sentiment, normalized tf-idf matrix form, $\alpha = 0.5$, word count threshold 11.

(b) driver 11, positive sentiment, mean matrix form, $\alpha = 0.5$, word count threshold 16.

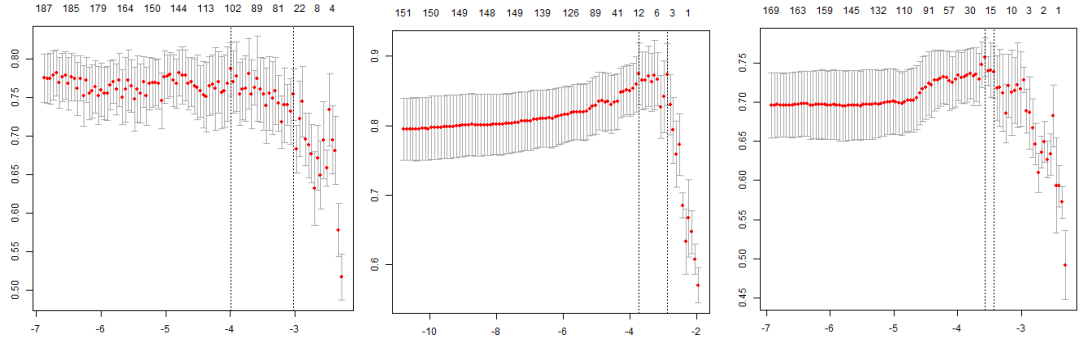
(c) driver 13, present sentiment, normalized tf-idf matrix form, $\alpha = 0.75$, word count threshold 6.



(d) driver 9, negative sentiment, indicator matrix form, $\alpha = 1$, word count threshold 21.

(e) driver 2, positive sentiment, document term matrix form, $\alpha = 0.5$, word count threshold 11.

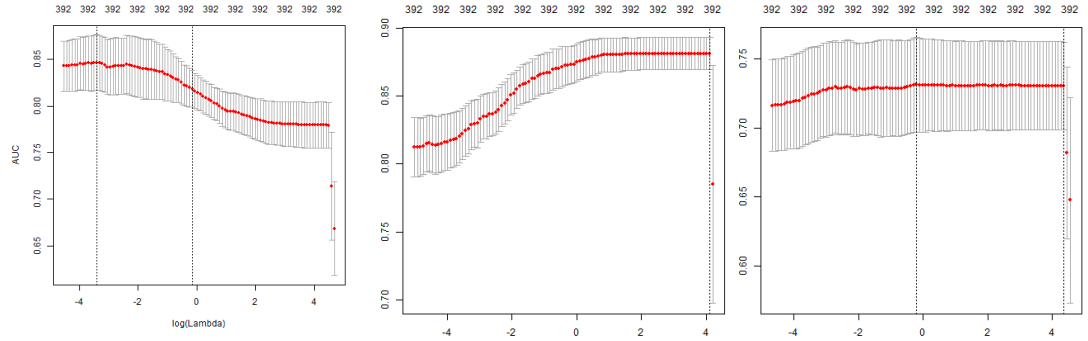
(f) driver 14, present sentiment, mean matrix form, $\alpha = 0.75$, word count threshold 11.



(g) driver 5, negative sentiment, indicator matrix form, $\alpha = 0.75$, word count threshold 1.

(h) driver 12, present sentiment, indicator matrix form, $\alpha = 1$, word count threshold 11.

(i) driver 9, present sentiment, document term matrix form, $\alpha = 1$, word count threshold 6.

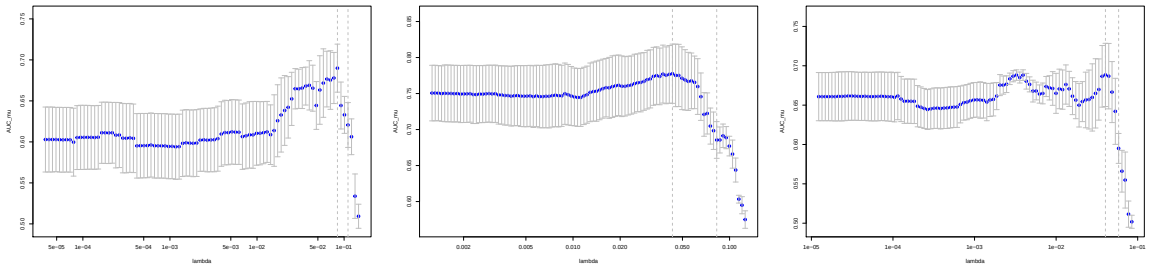


(j) driver 7, positive sentiment, indicator matrix form, $\alpha = 0$, word count threshold 11.

(k) driver 12, negative sentiment, tf-idf matrix form, $\alpha = 0$, word count threshold 11.

(l) driver 7, negative sentiment, normalized tf-idf matrix form, $\alpha = 0$, word count threshold 11.

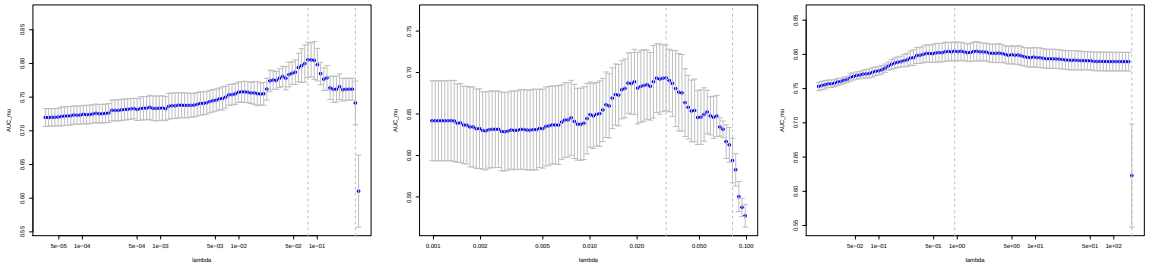
Figure 7.14: Plot of AUC estimates for models found for 100 different values of λ for various Elastic net models. On the vertical axis the AUC estimate is shown, on the lower axis the value of $\log(\lambda)$. The top axis shows the number of non-zero variables in the model associated with the corresponding λ on the lower axis. In all cases, a 5-fold cross validation was used and all results were obtained using the $n775$ data set. Results are shown for various options for: driver number, sentiment, matrix form , elastic net α (α) and word count threshold. An Elastic net model with $\alpha = 1$ is a LASSO, with $\alpha = 0$ is a Ridge regression.



(a) driver 6, normalized tf-idf matrix form, α = 0.5, word count threshold 11.

(b) driver 13, normalized tf-idf matrix form, α = 0.75, word count threshold 6.

(c) driver 9, indicator matrix form, α = 1, word count threshold 21.



(d) driver 2, document term matrix form, α = 0.5, word count threshold 11.

(e) driver 5, indicator matrix form, α = 0.75, word count threshold 1.

(f) driver 7, indicator matrix form, α = 0, word count threshold 11.

Figure 7.15: Plot of AUC_μ estimates for models found for 100 different values of λ for various Elastic net models. On the vertical axis the AUC_μ estimate is shown, on the lower axis the value of $\log(\lambda)$. The top axis shows the number of non-zero variables in the model associated with the corresponding λ on the lower axis. In all cases, a 5-fold cross validation was used and all results were obtained using the $n775$ data set. Results are shown for various options for: driver number, matrix form , elastic net α (α) and word count threshold. An Elastic net model with $\alpha = 1$ is a LASSO, with $\alpha = 0$ is a Ridge regression.

7.3.3. Number of folds used in determining λ

As mentioned in chapter 5, a cross validation is used to determine the value of λ used in LASSO and other Elastic net models (including Ridge regression). The number of folds used in this cross validation can be varied, but common choices are a 5-fold or 10-fold cross validation. Use more folds, and your test sets become extremely small, which can lead to unrepresentative results. Use fewer folds, and you are using a relatively small part of your data to train on, not using your model to its full potential. Regardless of which performs better, we need to choose the 5-fold though, because of the same problem that occurred when discussing the choice of the k. When performing the 10-fold cross validation, there are not for all drivers enough 1s (or Ps and Ns) for every test fold. Then it is not possible to use a 10-fold CV to train the Elastic net models.

This is quite a shame though. If we run all 4 Elastic net models with a 10-fold CV to determine the λ for all choices of word count threshold and matrix form, we find that use of the 10-fold CV gives better performing models. This can be seen by running for both 5-fold and 10-fold cross validations, for only those drivers for which we can obtain results using both methods. If we then plot the main effect and interaction effects, we find:

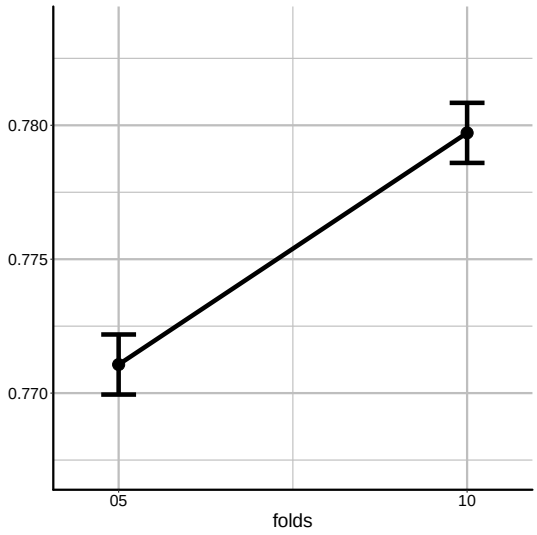
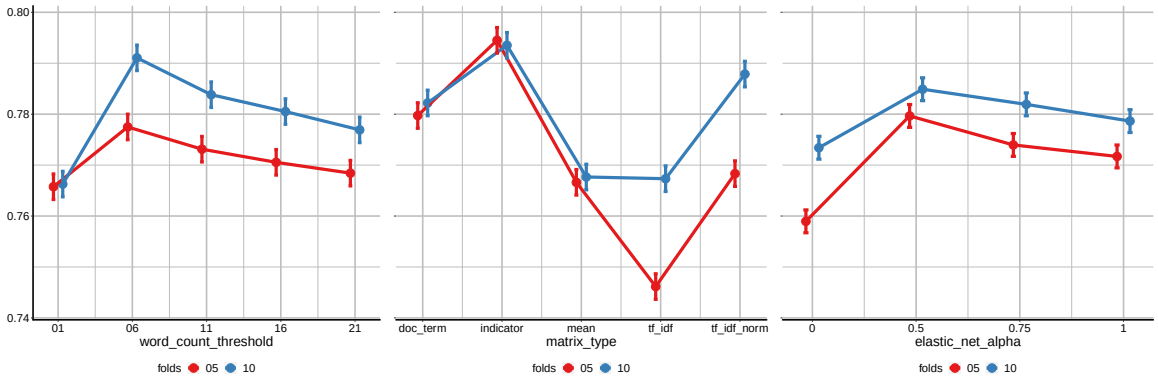


Figure 7.16: Main effects plot of the use of the number of folds in the cross validation for determining the λ in Elastic net models



(a) Interaction effect of word count threshold vs. number of folds.

(b) Interaction effect of matrix form vs. number of folds.

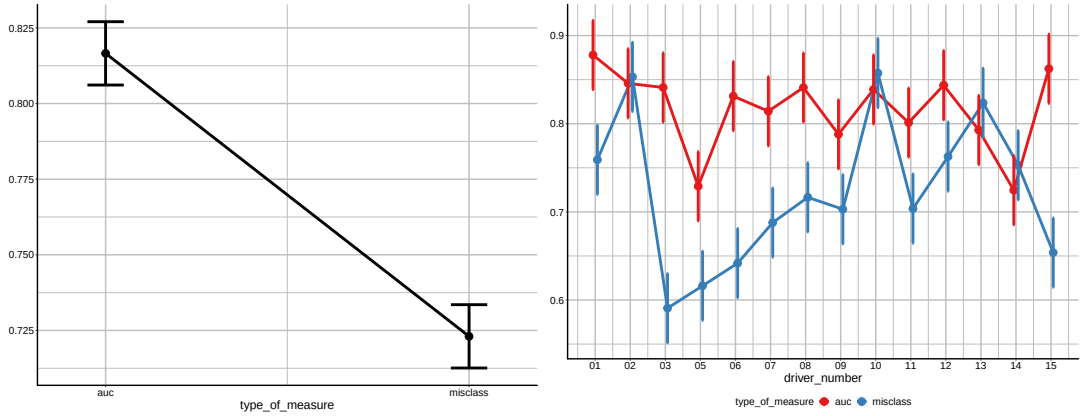
(c) Interaction effect of Elastic net model vs. number of folds.

Figure 7.17: Interaction effects plots of the number of folds with word count threshold, matrix form and Elastic net model. On the y-axis is the AUC value. On the x-axis is displayed the (first) varying parameter. The displayed confidence intervals are calculated as 1.96 × SE.

We thus see that it seems that it seems generally better to use a 10-fold cross validation. However, we will then not be able to obtain results for as many drivers as we do when using a 5-fold cross validation to determine λ. Because we consider it more important to use as many drivers as possible in our analysis, we choose for the 5-fold cross validation to determine λ anyway.

7.3.4. Evaluation measure

Now that we have dismissed the possibility of using different numbers of folds to determine the λ in Elastic net models, we use this moment to dismiss another factor. For the evaluation of our results, we introduced the AUC (in the binary setting) and AUC_μ. In chapter 4, we also briefly discussed the misclassification error as a possible evaluation method. In determining the λ, we have always sought for the model found using the λ that gives the highest AUC value. We could also choose the λ that does not necessarily give the highest AUC value, but does give the lowest misclassification error. Let us train a set of models using both the AUC and the misclassification error, using the same parameter settings. We find the following results:



(a) Effects plot for the evaluation measure.

(b) Effects plot for the driver numbers vs. the evaluation measure.

Figure 7.18: Effects plots of the type of evaluation measure (AUC and misclassification error) used in determining the λ in the LASSO model. On the y-axis is the AUC value. On the x-axis in the first plot we have the evaluation measure used. In the second plot, we have the driver numbers on the x-axis. The displayed confidence intervals are calculated as $1.96 \times \text{SE}$. The red line describes the models trained using the AUC, the blue line using the misclassification error. Result was obtained using the $n775$-data set, predicting present, positive and negative sentiment. Parameter settings used are document-term matrix form, indicator form and mean form and word count threshold 1.

Of course, models trained on the misclassification error have a lower AUC value than models trained on the AUC (the difference is 0.093). No surprises there. The reason for adding the second plot, is to understand what is really happening here. For some driver numbers, the measured AUC value is fairly close for both the models trained on AUC and the models trained on misclassification error. For other driver numbers the difference is rather large. The reason for this is the imbalance between 1's and 0's in the data set. For some combinations of driver and sentiment this imbalance is more severe than for others, but for none is there anything close to a 50-50 balance.

By letting the model train on misclassification error, the λ parameter for the model is chosen such that the model gives the most correct predictions when setting a certain threshold t. Take driver number 3, "Design", for example. Out of the 775 reviews in the data set, 33 reviews mention the driver, of which 16 with positive sentiment and 17 with negative sentiment. Let us aim to predict positive sentiment. When training on misclassification error one of two things will happen. The first option: a value for λ gets chosen such that the model predicts only 0's, which yields $1 - \frac{16}{775} = 0.98$ as the rate of correct predictions. The AUC value will be 0.5 and the model cannot distinguish 1's from 0's. The second option: The reviews mentioning 'Design' with positive sentiment are so distinct from the other reviews in the data set, that the 0.98 correct rate can be beaten by a model that effectively distinguishes 1's from 0's. As one might guess, the last option is not realistic. By letting the model train on AUC, a different λ and thus different model is selected, where the chosen λ is the one in the series giving the model that yields the highest AUC value. This model may not give the highest fraction of correct predictions, but it will actively try to distinguish 1's from 0's. As we are not interested in models predicting mostly 0's, the AUC is preferred in this thesis. Note that the measured AUC value for the models trained for driver number 3 using misclassification error do not actually score an average AUC of 0.5 though. Note that the point is an average over the different sentiments, including present in which case there is slightly less imbalance in 0's and 1's. Apparently the reviews discussing 'Design' can actually stand out enough to sometimes be predicted as a 1, even when the model has been trained on misclassification error.

For driver number 10, 'shaving closeness', models trained using the misclassification error score just as well as models trained using the AUC. The reason for this is again found in the number of reviews discussing this driver. Out of the 775 reviews in the data set, 270 mention 'shaving closeness', of which 158 with positive sentiment and 112 with negative sentiment. Thus, when training a model on the misclassification error, i.e. searching for a λ such that the resulting model gives the highest fraction of correct predictions at a certain threshold value, only predicting 0's does not give such a high fraction of correct predictions as it did for driver number 3. In this situation, a value of λ gets picked such that the resulting model does actively try to distinguish 1's from 0's. A model trained on AUC will also try to do this, and thus the model trained on misclassification error and the model trained on AUC will be close in their performance, and possibly even

be the same model (i.e. in both cases the same value was chosen for the λ parameter).

Because we in general do suffer from this class imbalance, we opt to train all elastic net models in the rest of this thesis using the AUC. For the same reason as why we reject the models trained on misclassification error, we will also not use the misclassification error to evaluate models after they are trained (both Elastic net and random forests); we are more interested in models that distinguish 1's from 0's (and thus give a high AUC value), than those that give the most correct predictions. These models can be similar of course, but this is generally not the case.

Thus concludes our discussion of the redundant parameters. We have discussed how we choose them, and we will use these same choices when computing the results in Chapter 8.

8

Results

In this chapter, we will show and discuss the obtained results. In Chapter 7 we discussed how these results are obtained (see Figure 7.3) and how we evaluate and compare them. We have three parameters in our model that we actively vary. These are:

	Parameter	# of options	options
1.	matrix form	5	standard, mean, indicator, tf-idf, normalized tf-idf
2.	word count threshold	5	$1, 6, 11, 16, 21$
3.	model	6	Elastic net for $\alpha \in [0, 0.5, 0.75, 1]$, two options for random forest

Figure 8.1: All parameters that we will actively vary with when obtaining our results.

We will apply our model and generate results for all combinations of these 3 parameters. Then we still have 2 different data sets, one containing 15 and the other containing 17 drivers. Furthermore we differentiate between the binary setting and the multinomial setting. In the binary setting, we have 3 different types of sentiment (present, positive and negative) whereas in the multinomial setting we have only 1 partition of the data set. Then again, we do have different choices of the misclassification matrix A to consider in the multinomial setting.

Our main interest is which combination of the parameters in Figure 8.1 gives us the best model. Furthermore, for a possible application of these models it is very useful to know which type of sentiment is easier to predict. The same holds for the question as to which drivers are the easiest to predict. We cannot really influence these from a modelling point of view, but we do this out of interest of applying our model and gaining insight into our data sets.

We will first discuss the results for the binary setting, and then for the multinomial setting.

8.1. Results for the binary setting

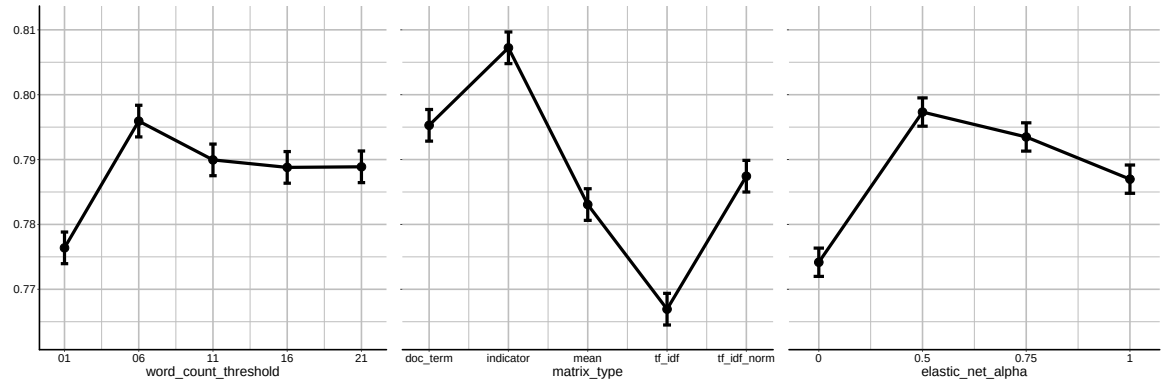
We will first discuss the results for the Elastic net models and then for the random forest models. After that, we will compare their performance. Results for the n775-data set and n1976-data set will be analyzed separately.

8.1.1. Elastic net models

We try four different Elastic net models: $\alpha = 0$ (Ridge regression), $\alpha = 0.5$, $\alpha = 0.75$ and $\alpha = 1$ (LASSO). All combinations of the parameters for matrix form and word count threshold are run. We run the models for present, positive and negative sentiment. This means that, for the n1976-data set, we run a total of 3 types of sentiment $\times$ 4 models $\times$ 5 word count thresholds $\times$ 5 matrix forms = 300 columns of results for 15 drivers similar to the results we saw in Chapter 7 (as in Figure 7.7). 15 drivers despite the n1976-data set containing 17 drivers, because for driver numbers 4 and 12 there are not enough non-NA annotated documents to generate

representative results for all 3 types of sentiment. Therefore, these are left out of this analysis. An ANOVA model is built around the results, and an estimate of the variance is obtained from this model to build confidence intervals around the values that we will plot. We will further supplement the number of AUC estimates on which plotted values are based, to give the reader more of an idea what they are looking at. Statistics obtained through ANOVA are not mentioned, unless there are insignificant factors or insignificant interactions between factors.

Thus, we can now look at the results obtained using the $n1976$-data set. We first discuss the main effects plots (plotted in Figure 8.2) and then the interaction effects plots (plotted in Figure 8.3).



(a) Effect of word count threshold. (b) Effect of matrix form. (c) Effect of Elastic net model.

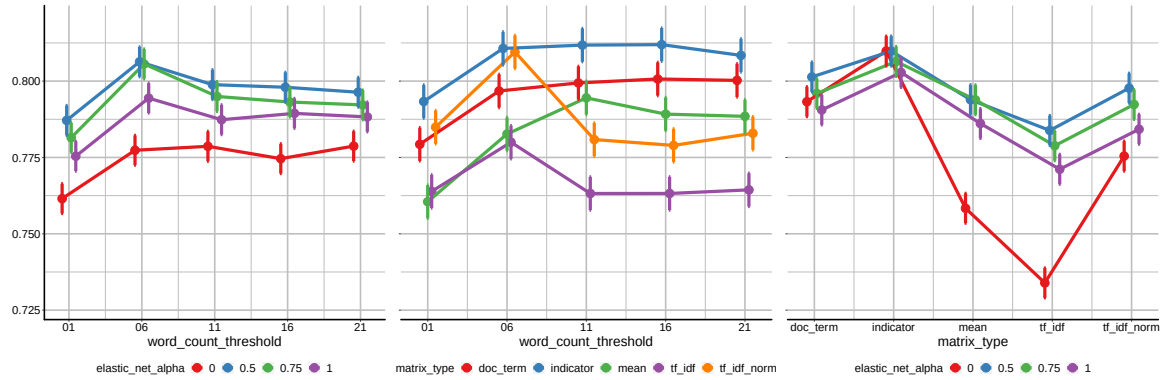
Figure 8.2: Effects plot for the effect of the matrix form, word count threshold and Elastic net α. On the y-axis is the AUC value, on the x-axis the varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times$ SE. Results were obtained using the $n1976$-data set running the models for all sentiments. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

When it comes to the variability of these points, the points in Figures 8.2a and 8.2b are averages over $300 \times 15/5 = 900$ AUC estimates (and again, these are the result of a 5-fold cross validation), and the points in Figure 8.2c are averages over $300 \times 15/4 = 1125$ AUC estimates.

In Figure 8.2a we see that a threshold of 6 performs best. Thus, we see that we give less variables, compared to a word count threshold of 1 (i.e. no threshold), to the models and they perform better. An hypothesis that would explain this is that we are basically further regularizing the model by removing uncommon words from the data set. By removing words that appear so infrequently, we are preventing the models from overfitting on these infrequently used words. The hypothesis for why a word count threshold of 6 performs better than one of 11 (or higher) would then be that at that point we are throwing too many actually informative variables, leading to a lower AUC value.

In Figure 8.2b we see that for matrix form the indicator form, where variables $x_1, \ldots, x_V$ take values 1 (if a review uses the word) or 0 (if it does not), performs best. A possible reason for the low performance of the mean and the tf-idf form, and to a lesser extent the normalized tf-idf form, is the long length of some of the reviews in the data set. Let us illustrate with a small example. Say that we are trying to predict whether a review mentions the driver 'shaving irritation' or not (i.e. present sentiment) and we use a LASSO model. The model estimates coefficients $\beta_1, \ldots, \beta_V$ and lets many shrink to 0. One coefficient that is non zero and is estimated as being fairly large is β_r, the coefficient 'rash' (as in, an irritation of the skin), and is thus considered as having predictive value for predicting whether or not a review mentions 'shaving irritation'. Let i be a review containing the word 'rash' such that $x_{i,r} \neq 0$ and let i contain a large number of words. When the value for $x_{i,r}$ is then determined through the use of the mean or tf-idf matrix form, the value found will be fairly low, because we divide by the total number of words in i which we assumed is high. The result is that, despite β_r being estimated as non-zero, the low value for $x_{i,r}$ makes the estimated probability of document i mentioning 'shaving irritation' rather low. The document matrix form and indicator form do not suffer from this problem, and this may be why we find higher AUC values for these forms. In this we assumed a LASSO was used for variable selection. Elastic net with $\alpha = 0$ (i.e. Ridge regression) does not let the estimates for coefficients shrink to 0 when they are considered irrelevant, but would still give 'rash' a higher estimated coefficient than others, thus the above hypothesis also generalizes to Ridge regression.

Lastly, we see in Figure 8.2c that an Elastic net model with $\alpha = 0.5$ performs best, while $\alpha = 0$ (Ridge regression) performs worst. The latter is in line with what we expect. Ridge regression does not perform variable selection, and we expect only a few of the variables to actually be relevant for predicting the driver scores. Thus Ridge is expected to perform worse. Why does Elastic net perform better for $\alpha = 0.5$ than for $\alpha = 0.75$ or $\alpha = 1$ (LASSO) though? This is a hard question to answer from the theory. All these options perform variable selection and are suited to deal with large numbers of variables. Zou and Hasty mention though that Elastic net often outperforms the LASSO in terms of prediction accuracy. [18]. Then between, $\alpha = 0.5$ and $\alpha = 0.75$, we say that one should outperform the other. Equal performance is almost an impossibility. We will now look at the interaction plots.



(a) Interaction effect of word count threshold vs. Elastic net model.

(b) Interaction effect of word count threshold vs matrix form.

(c) Interaction effect of matrix form vs Elastic net model.

Figure 8.3: Interaction effects plot for the effect of the matrix form, word count threshold and Elastic net α. On the y-axis is the AUC value, on the x-axis the first varying parameter for its respective effects plot. Results were obtained using the n1976-data set running the models for all sentiments. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

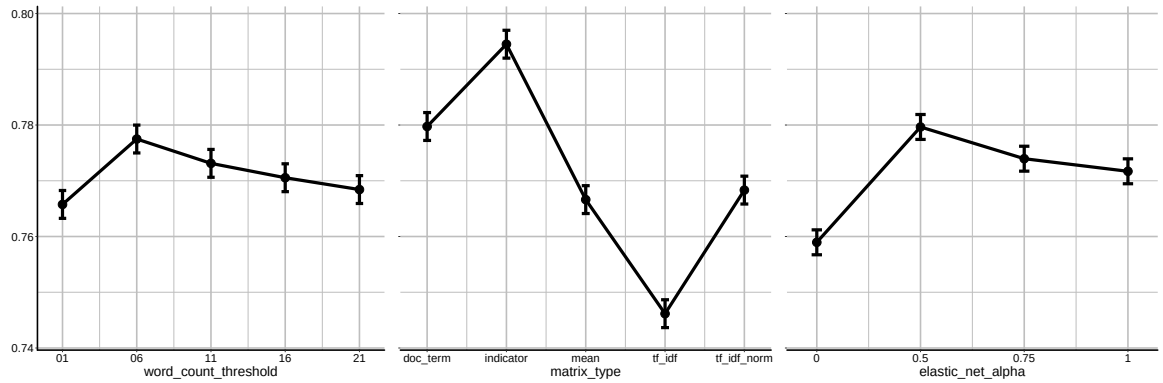
When it comes to the variability of these points, the points in Figure 8.3b are averages of $300 \times 15/(5 \times 5) = 180$ AUC estimates, and the points in Figures 8.3a and 8.3c are averages of $300 \times 15/(5 \times 4) = 225$ AUC estimates.

In Figure 8.3b we see that indicator matrix form performs best at all word count thresholds. Even more interesting, is that it seems to perform slightly better at word count thresholds 11 and 16 than it does at 6. They are really close though, so it may not be enough to conclude that they perform better (as we have to think of the variability in the estimates), but the performance is at least similar. For the document-term matrix form, we also see that it performs better at word count threshold 11 and 16. The tf-idf and normalized tf-idf clearly perform worse for word count thresholds higher than 6. Even more interesting, the normalized tf-idf matrix form, that performs worse at all other word count thresholds, performs almost as good as the indicator matrix form at a word count threshold of 6.

For the elastic-net models, we see that at all word count thresholds, the $\alpha = 0.5$ model performs best, and the performance is best at a word count threshold level of 6. What is interesting is that the performance of the $\alpha = 0.75$ model reaches nearly the same performance at that point.

When it comes to the interaction between the matrix form and the Elastic net models, we see that the $\alpha = 0.5$ model dominates the others with the $\alpha = 0.75$ model coming really close. Interesting is that when using the indicator matrix form, $\alpha = 0$ (Ridge regression) gives the same performance as $\alpha = 0.5$, but its performance utterly collapses for the mean and tf-idf matrix forms. Up to us the task of finding an explanation for this behaviour.

Now that we have had the main effects in Figure 8.2 and the interaction effect plots in Figure 8.3 for the $n1976$-data set, we compute the same figures using the $n775$-data set and compare. We run a total of 3 types of sentiment × 4 model × 5 word count thresholds × 5 matrix forms = 300 columns of results for 14 drivers. 14 drivers, because driver number 4 in the data set does not contain enough non-NA annotated reviews to build generate representative result for all 3 types of sentiment. Therefore, this driver is left out of the analysis. The results for the main effects are shown in Figure 8.4.

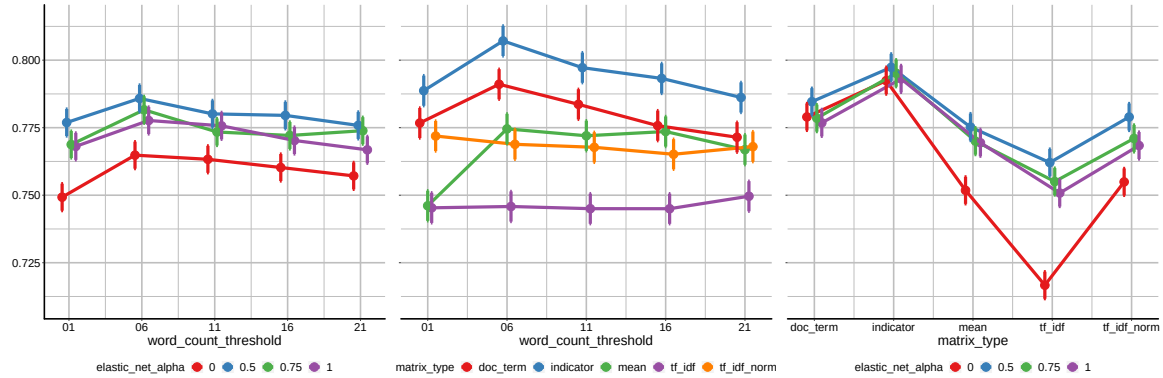


(a) Effect of word count threshold. (b) Effect of matrix form. (c) Effect of Elastic net model.

Figure 8.4: Effects plot for the effect of the matrix form, word count threshold and Elastic net α. On the y-axis is the AUC value, on the x-axis the varying parameter for its respective effects plot. The displayed confidence intervals are calculated as 1.96 × SE. Results were obtained using the 775-data set running the models for all sentiments. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

When it comes to the variability of these points, the points in Figures 8.4a and 8.4b are averages over $300 \times 14/5 = 840$ AUC estimates (and again, these are the result of a 5-fold cross validation), and the points in Figure 8.4c are averages over $300 \times 14/4 = 1050$ AUC estimates.

Comparing these with the results from Figure 8.2, where the $n1976$-data set was used, we see that here are hardly any differences. Some effects attain a different average AUC value of course, but this is to be expected. A different data set with different drivers was used. All values for the parameters are ranked in the same order in each effect plot, which is what matters most to us. The relative differences between some parameter settings show slight differences. Relative to other choices for the word count threshold, a word count threshold of 21 performs better in the results obtained for the $n1976$ data set than it does for the $n775$ data set for example. This effect can also be explained. The $n1976$ data set contains more reviews, and thus after all words get thrown away that are used 20 or less times, the $n1976$ data set has more variables that are retained. Still, this is only a slight difference. We move on to the interaction plots in Figure 8.5.



(a) Interaction effect of word count threshold vs. Elastic net model.

(b) Interaction effect of word count threshold vs matrix form.

(c) Interaction effect of matrix form vs Elastic net model.

Figure 8.5: Interaction effects plot for the effect of the matrix form, word count threshold and Elastic net α. On the y-axis is the AUC value, on the x-axis the first varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times$ SE. Results were obtained using the n775-data set running the models for all sentiments. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

When it comes to the variability of these points, the points in Figure 8.3b are averages of $300 \times 14/(5 \times 5) = 168$ AUC estimates, and the points in Figures 8.3a and 8.3c are averages of $300 \times 15/(5 \times 4) = 210$ AUC estimates.

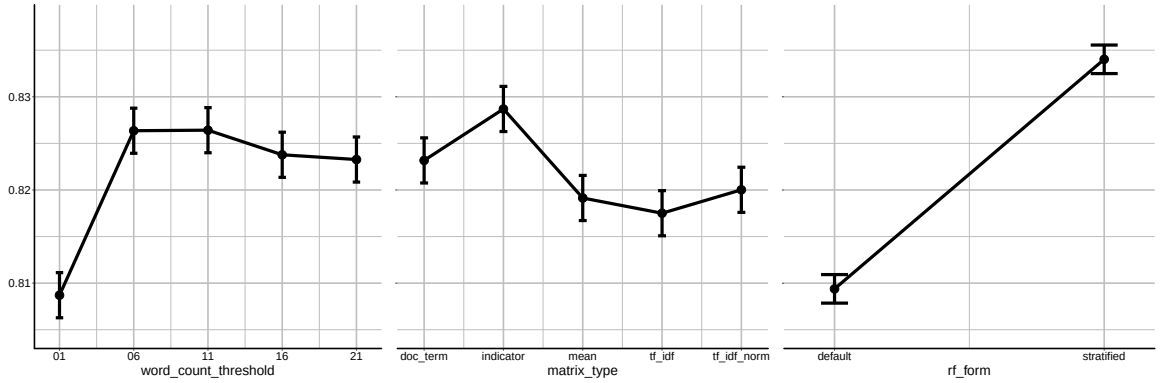
Now, look at the interaction plots in Figure 8.5. Overall, we see that the indicator matrix form dominates the other matrix forms at all word count thresholds in the top picture. In the bottom two pictures, Elastic net with $\alpha = 0.5$ performs better at every word count threshold level and for every matrix form. This is the same result as we saw in the results obtained for n1976-data set. When we look further, there are some differences though.

For example, for the n1976-data set, we saw in Figure 8.3 that the performance of the normalized tf-idf matrix form was very close to the performance of the indicator matrix form at a word count threshold level of 6 (which was the best at that threshold). In the top picture of Figure 8.5 we see that this is not the case, and the normalized tf-idf is outperformed by the mean matrix form and document-term matrix at a word count threshold of 6. Furthermore, we see in the same top picture in Figure 8.3 that for example the document-term matrix seems to perform only better as the word count threshold increases in the n1976-data set, whereas this does not seem to be the case here for the n775-data set.

All in all, we can conclude that to a large extent the effect of changing the parameter settings is the same between the n1976-data set and the n775-data set, only considering Elastic net models.

8.1.2. Random forest models

We run two different types of random forest models, the 'default option' and the 'stratified option', as described in Chapter 6. All combinations of the parameters for matrix form and word count threshold are run. We run the models for present, positive and negative sentiment. This means that, for the n1976-data set, we run a total of 3 types of sentiment × 2 models × 5 word count thresholds × 5 matrix forms = 150 columns of results for 15 drivers. As we did for the Elastic net models, we will first discuss the main effects plots and then the interaction effects plots.

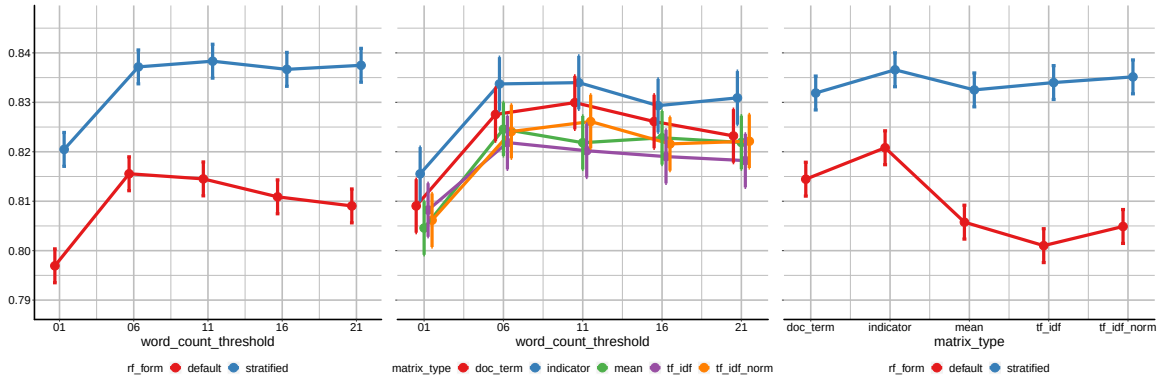


(a) Effect of word count threshold. (b) Effect of matrix form. (c) Effect of option for random forest.

Figure 8.6: Effects plot for the effect of the matrix form, word count threshold and random Forest models. On the y-axis is the AUC value, on the x-axis the varying parameter for its respective effects plot. The displayed confidence intervals are calculated as 1.96 × SE. Results were obtained using the n1976-data set running the models for all sentiments. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

When it comes to the variability of these points, the points in Figures 8.6a and 8.6b are averages over 150 × 15/5 = 450 AUC estimates (and again, these are the result of a 5-fold cross validation), and the points in Figure 8.6c are averages over 150 × 15/2 = 1125 AUC estimates.

In Figures 8.6a and 8.6b we see that for the main effects of the word count threshold and matrix form we have a similar effect using the random forest models as we found for the Elastic net models, for the n1976-data set (Figures 8.2a and 8.2b). At first sight, a word count threshold level of 6 and an indicator matrix form perform best. From Figure 8.6c, we get the first hand impression that the random forest with the stratified adaption performs best. Let us move to the interaction plots in Figure 8.7, and see if these can shed a different light on the results.



(a) Interaction effect of word count threshold vs. random forest model. (b) Interaction effect of word count threshold vs matrix form. (c) Interaction effect of matrix form vs random forest model.

Figure 8.7: Interaction effects plot for the effect of the matrix form, word count threshold and random forest model. On the y-axis is the AUC value, on the x-axis the first varying parameter for its respective effects plot. The displayed confidence intervals are calculated as 1.96 × SE. Results were obtained using the n1976-data set running the models for all sentiments. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

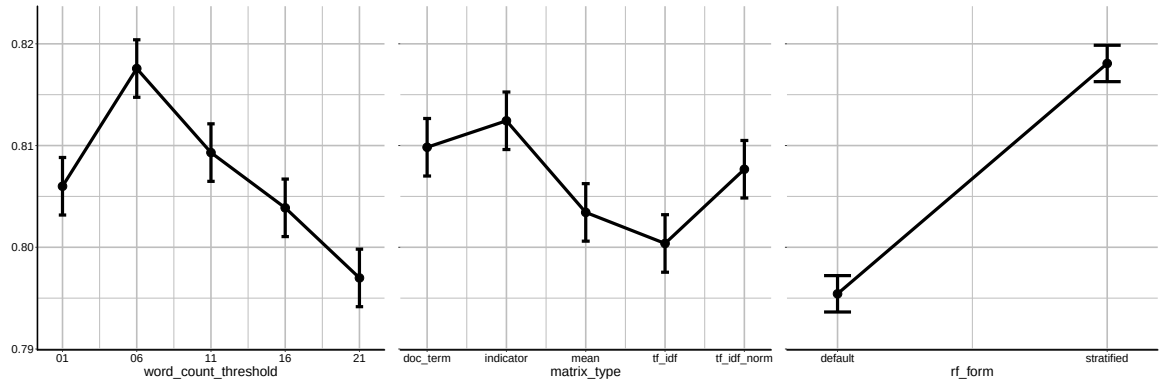
When it comes to the variability of these points, the points in Figure 8.7b are averages over $150 \times 15/(5 \times 5) = 90$ AUC estimates and the points in Figures 8.7a and 8.7c are averages over $150 \times 15/(5 \times 2) = 225$ AUC estimates (and again, these are the result of a 5-fold cross validation).

In Figure 8.7a we see an interesting difference between the usage of the default option random forest and the stratified option. For the default option, the effect of the word count threshold shows the same pattern as to what we saw earlier for the Elastic net models (Figure 8.3a, and also Figure 8.5a for the n775-data set). For these models, a word count threshold of 6 gave the best performance of the models, which then repeatedly went down for larger values of the word count threshold. For the stratified option, this seems different. The performance stays more or less the same and even seems to increase a bit for word count thresholds of 11 and 21. The difference is small though, thus given the limited number of AUC estimates these numbers are based on we cannot draw the conclusion that the effect of further increasing the word count threshold is truly beneficial. We can state though that there does at least not seem to be a decrease in the performance of our models, as a result of increasing the word count threshold levels beyond 6 when using the stratified option random forest models. It seems that when using the stratified option the loss of information from increasing the word count threshold, which would explain the earlier results, is nullified with regards to predicting the driver scores.

In Figure 8.7b we see the same pattern as we saw earlier using the Elastic net models (Figure 8.3b, and also Figure 8.5b for the n775-data set). The indicator matrix form performs best at all word count threshold levels.

In Figure 8.7c we again see an interesting difference between the usage of the default option random forest and the stratified option. For the default option, the indicator matrix form performs best and the tf-idf matrix form performs worst. We observed the same pattern when analyzing the Elastic net models (Figure 8.3c, and also Figure 8.5c for the n775-data set). However, the stratified options behaviour is different. The indicator matrix form still performs best, but the performance of the others is very close. Especially the performance of the normalized tf-idf matrix is almost equal to that of the indicator matrix form.

We now move on to the results obtained by applying the random forest methods to the n775-data sets. We run a total of 3 types of sentiment × 2 models × 5 word count thresholds × 5 matrix forms = 150 columns of results for 14 drivers. The main effects plots are shown in Figure 8.8.

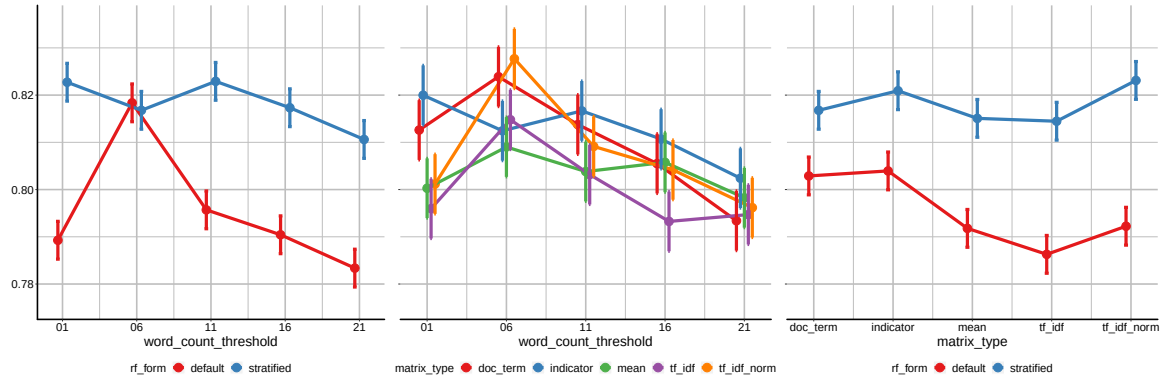


(a) Effect of word count threshold. (b) Effect of matrix form. (c) Effect of option for random forest.

Figure 8.8: Effects plot for the effect of the matrix form, word count threshold and Random Forest models. On the y-axis is the AUC value, on the x-axis the varying parameter for its respective effects plot. The displayed confidence intervals are calculated as 1.96 × SE. Results were obtained using the n775-data set running the models for all sentiments. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

When it comes to the variability of these points, the points in Figures 8.8a and 8.8b are averages over $150 \times 14/5 = 420$ AUC estimates (and again, these are the result of a 5-fold cross validation), and the points in Figure 8.8c are averages over $150 \times 14/2 = 1050$ AUC estimates.

To a large extent, we see the same effects occur as we did for the n1976-data set (as in Figure 8.6). We see that, again, on average the stratified option outperforms the default version and the word count threshold of 6 performs and indicator matrix form perform best. The main effect of varying the word count threshold for the n775-data set, is quite different to the effect we saw when using random forest methods for the n1976-data set (as shown in Figure 8.6a). We saw the same difference in effect occur between the data sets for the elastic net models though (see Figures 8.2a and 8.4b) and we explained the difference by the size of the data set. Let us look at the interaction effect plots in Figure 8.9.



(a) Interaction effect of word count threshold vs. random forest model. (b) Interaction effect of word count threshold vs matrix form. (c) Interaction effect of matrix form vs random forest model.

Figure 8.9: Interaction effects plot for the effect of the matrix form, word count threshold and random forest model. On the y-axis is the AUC value, on the x-axis the first varying parameter for its respective effects plot. The displayed confidence intervals are calculated as 1.96 × SE. Results were obtained using the n775-data set running the models for all sentiments. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

When it comes to the variability of these points, the points in Figure 8.9b are averages over $150 \times 14/(5 \times 5) = 84$ AUC estimates and the points in Figures 8.9a and 8.9c are averages over $150 \times 14/(5 \times 2) = 210$ AUC estimates

(and again, these are the result of a 5-fold cross validation).

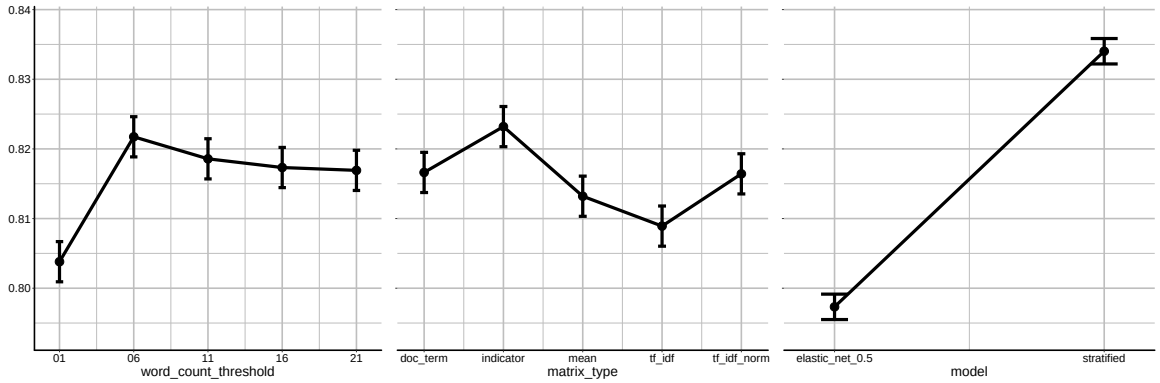
Analyzing Figure 8.9a, we see something odd. The performance of the stratified option for a word count threshold of 6 has slightly gone down compared to word count thresholds of 1 and 11, while the performance of the default option reaches its peak there, such that the performance of both random forest methods is more or less equal. The default option displays the 'regular' behaviour (the effect also occurs for the $n1976$-data set in Figure 8.7a and is also visible in the elastic net models as shown in Figures 8.3a and 8.5a). The behaviour of the stratified option is much out of the ordinary though. We have not seen a similar effect in earlier results, for random forest or elastic net models, in either of the data sets. It is particularly strange that the performance of the stratified option does not reach a peak, but is about equal for word count thresholds of 1 and 11. We expect, also in an interaction plot, that a low word count threshold may lead to overfitting on the data, and a high word count threshold may lead to throwing away valuable information, meaning that there is an optimum for the word count threshold somewhere in the middle. However, this does not seem to be the case here. Of course, we always have to think about the fact that there is some randomness in these points, but it seems too easy to blame this strange effect entirely on that.

Figure 8.9b also reveals some interactions we have not seen before. All of a sudden, the document term matrix form and especially the normalized tf-idf matrix form perform much better than the indicator matrix form at a word count threshold value of 6. It looks like the indicator matrix form performs worse at a word count threshold value of 6 than it does at word count threshold values of 1 and 11 (the same effect as the stratified option seems to be suffering from), however the document term and normalized tf-idf matrix forms do not seem to suffer from this problem, and achieve their peak performance.

Lastly, there is another quite interesting interaction in Figure 8.9c. We see that the for the stratified option, the normalized tf-idf matrix achieves a higher average AUC value than the indicator matrix form. In the results for the random forests using the $n1976$-data set (see Figure 8.7c) we also saw the normalized tf-idf matrix form achieve fairly good results, and to a lesser extent we also saw this for the elastic net models (Figures 8.3c and 8.5c). The combination of the normalized tf-idf and the stratified random forest, seems to be enough to now 'push' the results a little further up, giving good results in both Figures 8.7c and 8.9c.

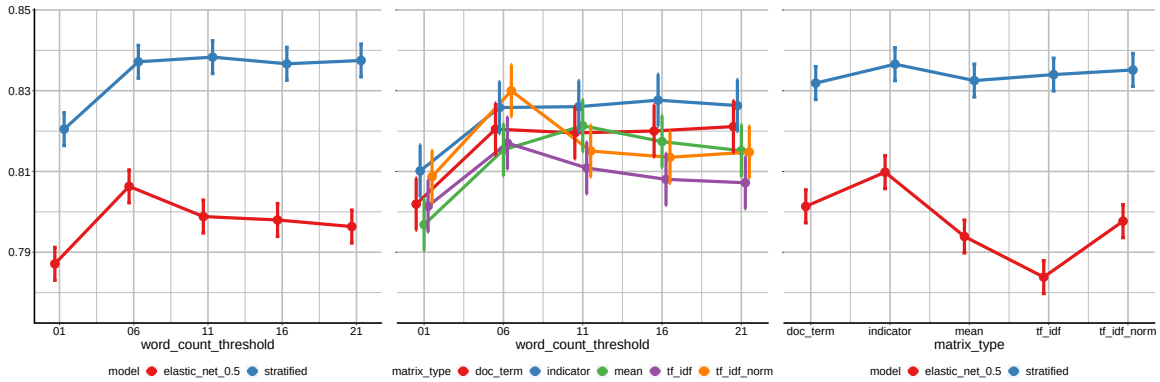
8.1.3. Comparing random forest and Elastic net models

Given our previous results, we select the Elastic net with $\alpha = 0.5$ and the random forest with the stratified option as the two most promising models from the set of elastic net and random forest models respectively. We compute the effects plots as before. Let us start with the $n1976$-data set. The main effects plots are shown in Figure 8.10 and the interaction effects plots in Figure 8.11.



(a) Effect of word count threshold. (b) Effect of matrix form. (c) Effect of model.

Figure 8.10: Effects plot for the effect of the matrix form, word count threshold and stratified option random forest and Elastic net with $\alpha = 0.5$ as models. On the y-axis is the AUC value, on the x-axis the varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times$ SE. Results were obtained using the $n1976$-data set running the models for all sentiments. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.



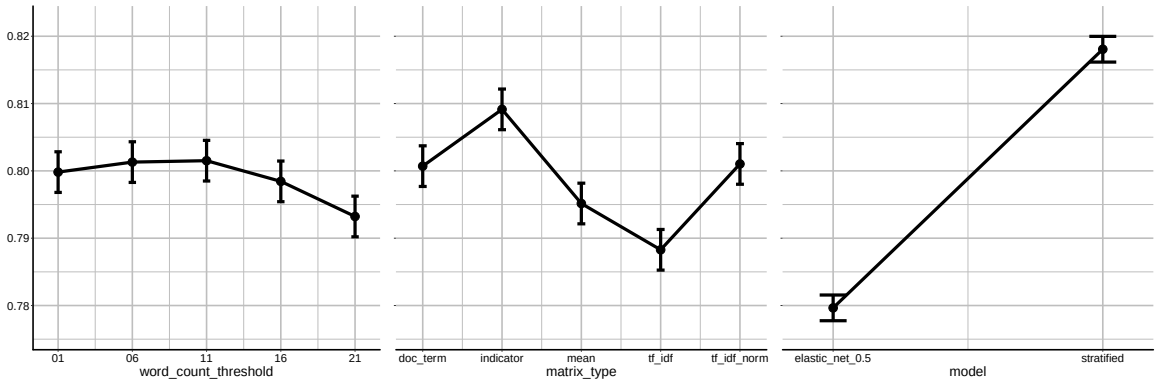
(a) Effect of word count threshold vs model. (b) Effect of word count threshold vs matrix form. (c) Effect of matrix form vs. model.

Figure 8.11: Interaction effects plot for the effect of the matrix form, word count threshold and stratified option random forest and Elastic net with $\alpha = 0.5$ as models. On the y-axis is the AUC value, on the x-axis the first varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times$ SE. Results were obtained using the $n1976$-data set running the models for all sentiments. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

When it comes to the variability of the points in the main effects graphs, the points in Figures 8.10a and 8.10b are averages over $150 \times 15/5 = 450$ AUC estimates (and again, these are the result of a 5-fold cross validation), and the points in Figure 8.10c are averages over $150 \times 15/2 = 1125$ AUC estimates. For the interaction effects graphs, the points in Figures 8.11a and 8.11c are averages over $150 \times 15/(5 \times 2) = 225$ AUC estimates (and again, these are the result of a 5-fold cross validation), and the points in Figure 8.11b are averages over $150 \times 15/(5 \times 5) = 90$ AUC estimates.

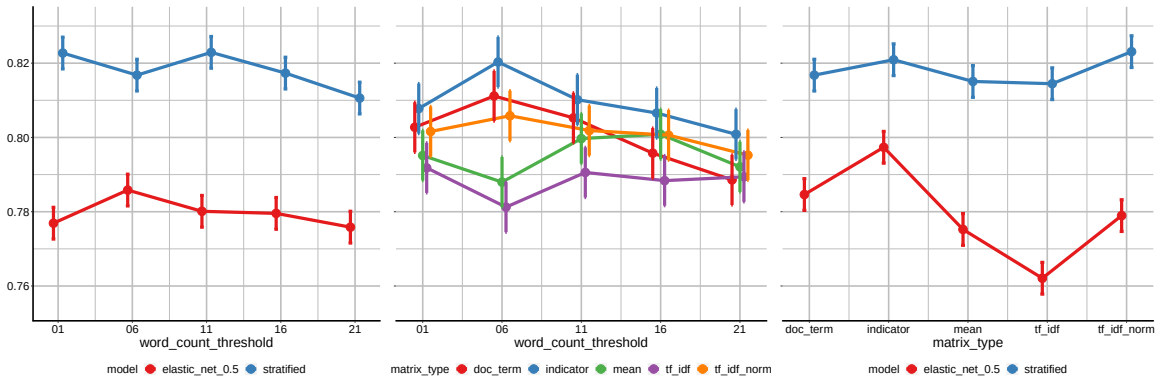
First Figure 8.10. We have seen the effects described by Figures 8.10a and 8.10b before. From Figure 8.10c we observe that overall the stratified random forest model performs better than the Elastic net with $\alpha = 0.5$. Now take a look at the interaction effects.

In Figure 8.11a, we see that the stratified random forest performs better than the Elastic net model with $\alpha = 0.5$ at all word count thresholds. The interaction between the stratified model and word count threshold is interesting, but has been discussed before (see Figure 8.7a and corresponding discussion). In Figure 8.11b we see the normalized tf-idf matrix form for a word count threshold of 6 perform best. This is interesting, because we did not see this happen in earlier results, when looking at all Elastic net or random forest models (Figures 8.3b and 8.7b). Apparently, at least one or maybe even both of these models (the stratified random forest and Elastic net with $\alpha = 0.5$) perform well for the combination of tf-idf matrix and a word count threshold of 6. This analysis is based on a rather small number of points though. In Figure 8.11c, we see that the stratified option random forest outperforms the elastic net model with $\alpha = 0.5$ for all matrix forms. We will now look at the results obtained using the n775-data set.



(a) Effect of word count threshold. (b) Effect of matrix form. (c) Effect of model.

Figure 8.12: Effects plot for the effect of the matrix form, word count threshold and stratified option random forest and Elastic net with $\alpha = 0.5$ as models. On the y-axis is the AUC value, on the x-axis the varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times$ SE. Results were obtained using the n775-data set running the models for all sentiments. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.



(a) Effect of word count threshold vs model. (b) Effect of word count threshold vs matrix form. (c) Effect of matrix form vs. model.

Figure 8.13: Interaction effects plot for the effect of the matrix form, word count threshold and stratified option random forest and Elastic net with $\alpha = 0.5$ as models. On the y-axis is the AUC value, on the x-axis the first varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times$ SE. Results were obtained using the n775-data set running the models for all sentiments. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

When it comes to the variability of the points in the main effects graphs, the points in Figures 8.12a and 8.12b are averages over $150 \times 14/5 = 420$ AUC estimates (and again, these are the result of a 5-fold cross validation), and the points in Figure 8.12c are averages over $150 \times 14/2 = 1050$ AUC estimates. For the interaction effects graphs, the points in Figures 8.13a and 8.13c are averages over $150 \times 14/(5 \times 2) = 210$ AUC estimates (and again, these are the result of a 5-fold cross validation), and the points in Figure 8.11b are averages over $150 \times 14/(5 \times$

5) = 84 AUC estimates.

We see the same effect occur as that we see for the $n1976$-data set. The stratified random forest model performance is better than that of the Elastic net model with $\alpha = 0.5$ in all cases. We thus regard the stratified random forest as the superior model for the problem discussed in this thesis. Breiman, one of the first notable contributors to the study of random forests, has stated in later research that use of stratified sampling, as was done in this thesis, is a powerful technique to improve performance of random forest methods when having imbalanced data sets[2]. Explaining why the random forest than also performs better than the Elastic net model is a hard nut to crack though. The Elastic net model with $\alpha = 0.5$ is also a variable selection technique, whereas the random forest is not. This would imply that dropping the requirement of variable selection causes an increase in performance. However, one could argue that in a scenario as ours, where many variables play no role of importance, variable selection is actually required. Ridge regression, which is also not a variable selection technique, has a significantly lower performance than the other Elastic net models. This remains a difficult, but nonetheless interesting debate. For now, we conclude that the stratified random forest is the superior model.

From these plots, we can conclude that given the choice between the Elastic net with $\alpha = 0.5$ model and the stratified option random forest model, the latter seems preferable when looking at the resulting AUC values.

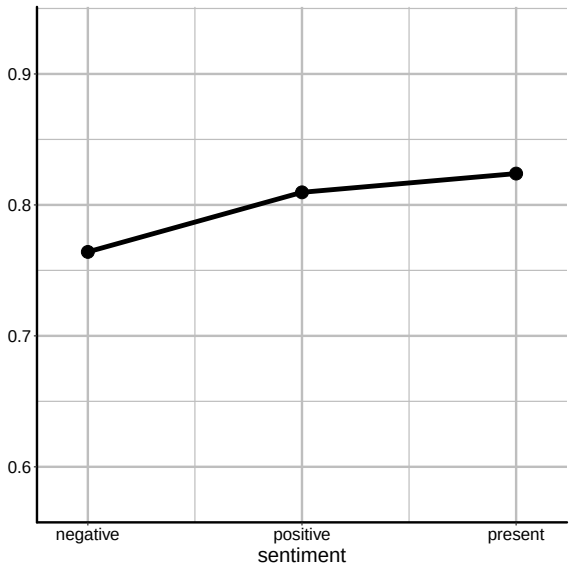
8.1.4. Effect of sentiment and driver number

Now take a step back to what we are trying to predict; driver scores for different sentiments. Positive sentiment may be easier to predict than negative sentiment, or the other way around. Furthermore, this can of course be different for every driver: for one driver positive sentiment might be easier to predict, and for another negative sentiment could be the easiest. We compare for both the $n1976$-data set and the $n775$-data set and create the effects plots. We do not visualise the error bars in these plots. This, because the scale on the y-axis will be such that the error bars will be hardly visible.

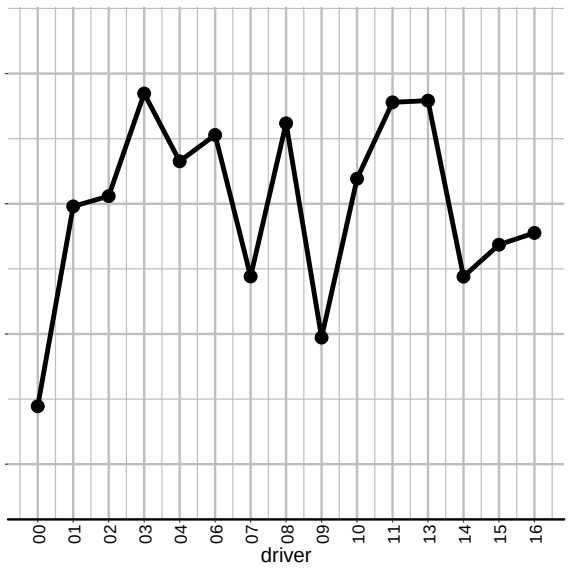
In Figure 8.14 we see the effects plots for for the $n1976$-data set. We see that within this data set, present sentiment is generally the easiest to predict, after that positive sentiment and then negative sentiment. For an explanation for this, we take a look back at Figure 2.3. It has been reprinted in Figure 8.15. We see that there are also more positive reviews than negative reviews in the $n1976$-data set. Thus, there is more data from the positive class to train on, and predicting for positive sentiment will generally give a more accurate result. Using this same reasoning, present sentiment gives the most accurate result. When detecting present sentiment, we aim to distinguish the non-NA driver scores from the NA driver scores. When detecting positive or negative sentiment, we aim to distinguish a subset of these non-NA driver scores from the rest, and there is thus less data in the minority class than when detecting present sentiment. Thus, the model generally performs worse when aimed at detecting negative sentiment.

Looking at the effects plot for the driver numbers, we also see some large differences. For some we reach an average AUC value of around 0.85, whereas for other we reach one far below 0.7. To an extent this also the result of the number of reviews discussing said driver available in data set as training data. For some this is not true at all though. For the driver number 0, there is a lot of data available but it has the lowest average AUC value of all driver numbers. In Section 2.1 we discussed though that this driver is odd, in that it characterises the general sentiment in the entire review, and is thus quite unlike the other drivers. Despite this, we still conclude that not all drivers adhere to the earlier described effect. For driver number 11 for example, there is quite a limited number of reviews discussing it available in the data set, but it scores quite high in AUC value. Apparently, reviews discussing driver 11 use words distinct enough from reviews that do not discuss driver 11, such that the models are able to score high AUC values for driver number 11. In the end, the case is different for every driver of course. Note that for some drivers no AUC value has been printed. As can be seen in Figure 8.15, there are drivers where the number of reviews discussing them is extremely limited. These have been removed from the analysis, as an effective AUC value could not be computed for these drivers.

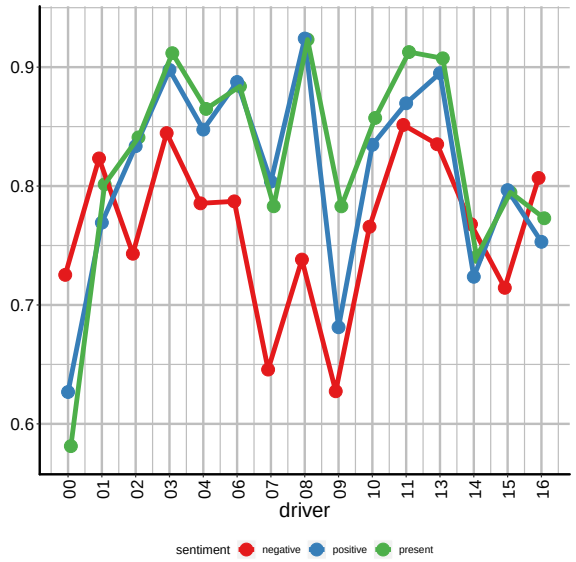
In Figure B.8a. We see that generally, we obtain lower AUC values for negative sentiment than we do for positive and present sentiment. Note that when there is more balance between the number of negative and positive reviews, the results are generally similar. Driver number 1 is an example of this effect. Generally there is an imbalance though. What part of the effect of positive sentiment being easier to predict than negative sentiment is caused by class imbalance and what part by the actual words/language? In the end, this is hard to judge. The results give a strong indication though that some drivers are inherently easier to predict than others (because of the language used), while other drivers are easier to predict merely because there are more reviews discussing them in the training data.



(a) Effect of sentiment.



(b) Effect of driver number.



(c) Interaction effect of driver number vs. sentiment.

0.	Shaving (general)
1.	During shaving – speed
2.	During shaving – comfort
3.	After shaving – irritation
4.	After shaving – closeness
5.	Cofo
6.	Sound
7.	Ease of use
8.	Cleaning
9.	Smart Clean
10.	Service & Delivery
11.	Accessories
12.	Features
13.	Charging & electronic power
14.	Design
15.	Value for money
16.	Issues & reliability

(d) Drivers corresponding to the driver numbers in the effect plots.

Figure 8.14: Effects plot for the effect of the sentiment and driver number and their interaction effects plot. On the y-axis is the AUC value, on the x-axis the varying parameter for its respective effects plot. Confidence intervals are not displayed. Results were obtained using the n1976-data set running elastic net models for all levels of α and both random forest options at all word count thresholds for all matrix types. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

When it comes to the variability of the points, the points in Figure 8.14a are averages over $450 \times 15/3 = 2250$ AUC estimates, the points in Figure 8.14b are averages over 450 AUC estimates, and the points in Figure 8.14c are averages over $450 \times 15/(15 \times 3) = 150$ AUC estimates.

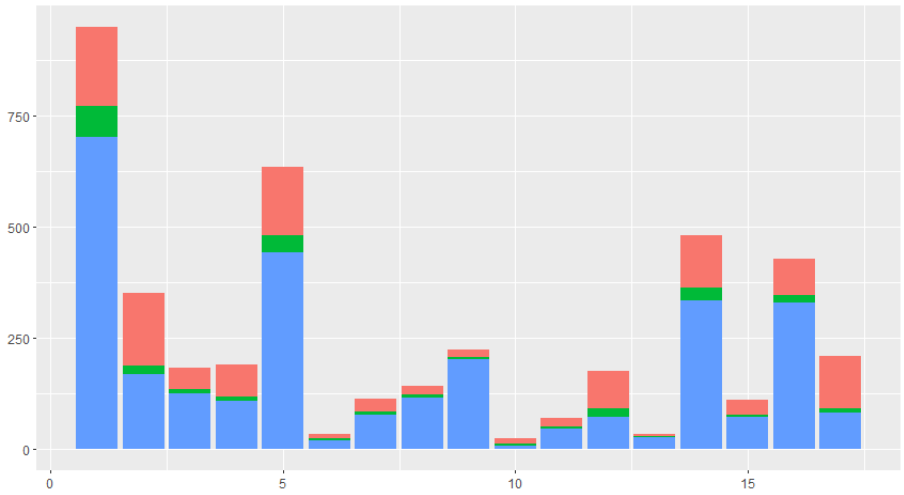


Figure 8.15: Histogram of the number of driver scores for every driver in the $n1976$ – data set. In blue the number of positive, in green the neutral and in red the negative driver scores.

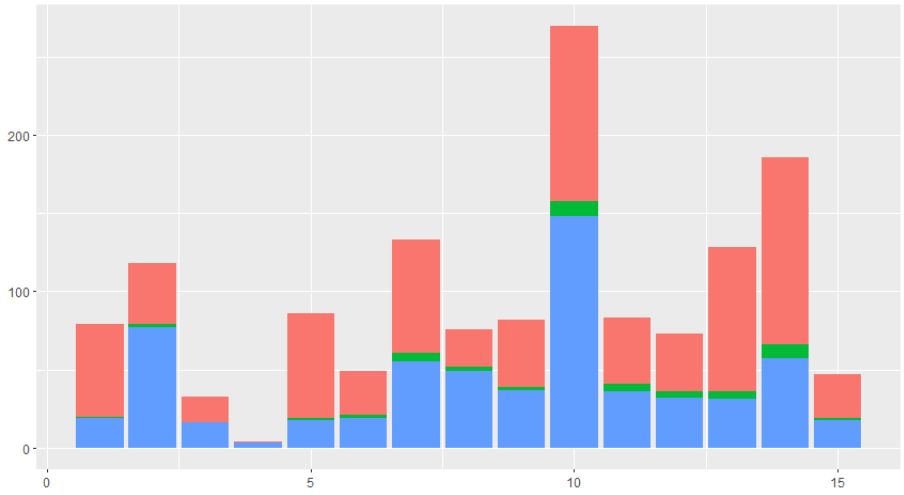
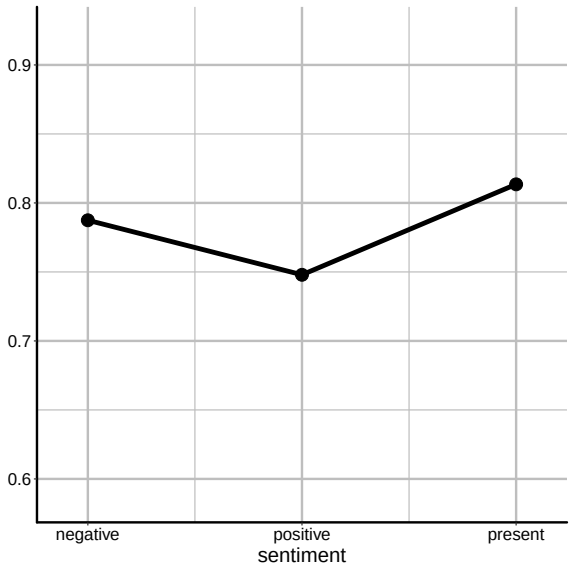
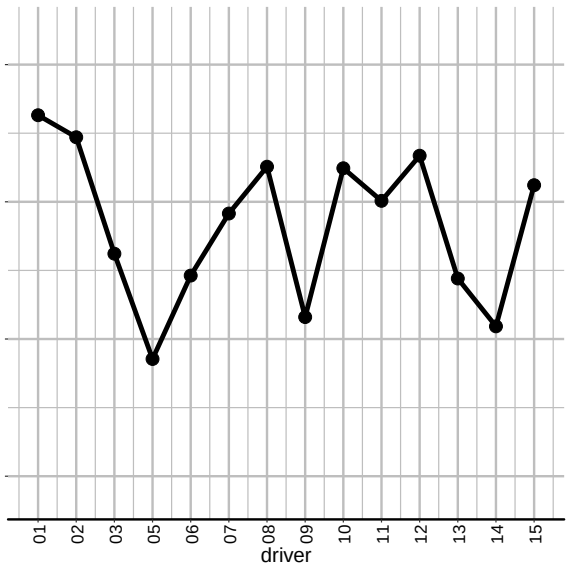


Figure 8.16: Histogram of the number of driver scores for every driver in the $n775$ – data set. In blue the number of positive, in green the neutral and in red the negative driver scores.

The same analysis that we did for the sentiment and driver numbers in the $n1976$-data set can be done for the $n775$-data set. Results are shown in Figure 8.17. Here we see the opposite result when it comes to sentiment that we saw for the $n1976$-data set. Negative sentiment is predicted more accurately than positive sentiment. A reason for this can be found in the same hypothesis that was used to explain the difference in the $n1976$: looking at Figure 8.16, we see that the we simply have a lot more negative reviews than positive reviews available. For the average AUC estimates for the individual drivers we also come to the same conclusion. Generally, we see that drivers for which more training data is available perform better. This is not always the case though. For driver number 1 there are not too many reviews available, but it still performs better than all others. This shows that the difference in results is not always caused by just the imbalance of the number of reviews annotated as a 1 versus the number of reviews annotated as a 0. Some drivers seem to be inherently easier to predict. A possible hypothesis for this is found in the fact that our models use the individual words written in the reviews as variables. Maybe people tend to use a smaller set of unique words when talking about topics related to certain drivers than they do when talking about topics related to others. The smaller the set of unique words used, the easier it is for our models to find the pattern of words used and driver (and sentiment) discussed. Looking at the interaction, we see that when there is no imbalance in sentiment for a review, positive sentiment can also perform well. This is for example the case for driver number 8.



(a) Effect of sentiment.



(b) Effect of driver number.

(c) Interaction effect of driver number vs. sentiment.

1. Accessories
2. Charging & Electronic power
3. Design
4. Features
5. Issues & Reliability
6. Service & Delivery
7. Value for money
8. cleaning
9. ease of use
10. shaving closeness
11. shaving comfort
12. shaving irritation
13. shaving rest
14. shaving speed
15. smart clean

(d) Drivers corresponding to the driver numbers in the effect plots.

Figure 8.17: Effects plot for the effect of the sentiment and driver number and their interaction effects plot. On the y-axis is the AUC value, on the x-axis the varying parameter for its respective effects plot. Confidence intervals are are not displayed. Results were obtained using the n775-data set running elastic net models for all levels of α and both random forest options at all word count thresholds for all matrix types. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

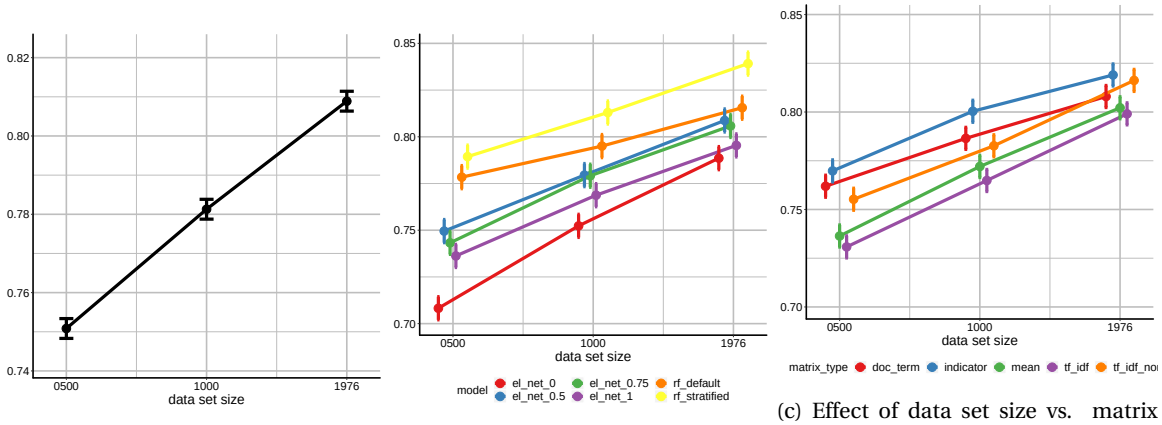
When it comes to the variability of the points, the points in Figure 8.17a are averages over $450 \times 14/3 = 2100$ AUC estimates, the points in Figure 8.17b are averages over 450 AUC estimates, and the points in Figure 8.17c are averages over $450 \times 14/(14 \times 3) = 150$ AUC estimates.

8.1.5. Effect of data set size

Another factor of interest is to us the size of data set. We generally presume that when given more data, our models perform better. Let us see if we can confirm this hypothesis, and if so, find out how large the effect of increasing the size of the data set can be. We have 1976 reviews in the $n1976$. We take this to be the maximum number of reviews we experiment, and we will also experiment with sampled subsets of these 1976 reviews of sizes 500 and 1000.

Normally, we choose a type of sentiment, matrix form, model, driver et cetera and run the model to find our result. Doing this repeatedly for all choices gives us the results as discussed in this Chapter. Thus, we also do this for the 3 different data set sizes. We did not experiment with the word count threshold and kept it constant at 11. We ran the model for all elastic net and random forest models and all matrix forms. It is important to note that given our choice of sentiment and driver, we did make sure that there were enough reviews being annotated as a 1 in the sampled subset of size 500/1000. Furthermore, we re-sampled the subsets repeatedly, not only for different drivers and sentiment, but also for different choices of matrix form and the model. This, to minimize the impact of the variance in sampling these subsets on our result.

The effects that interest us are of course the overall effect of increasing the size of the data set, but also the interactions between the size of the data set and the matrix form and model. Furthermore, we hypothesized before that some drivers are inherently easier to predict because of the language used when discussing them, and other drivers had more 'profit' of having more reviews discussing them in the training data. Thus, the effect of the data set size vs. individual drivers in particular is interesting to observe.



(a) Effect of data set size.

(b) Effect of data set size vs. model.

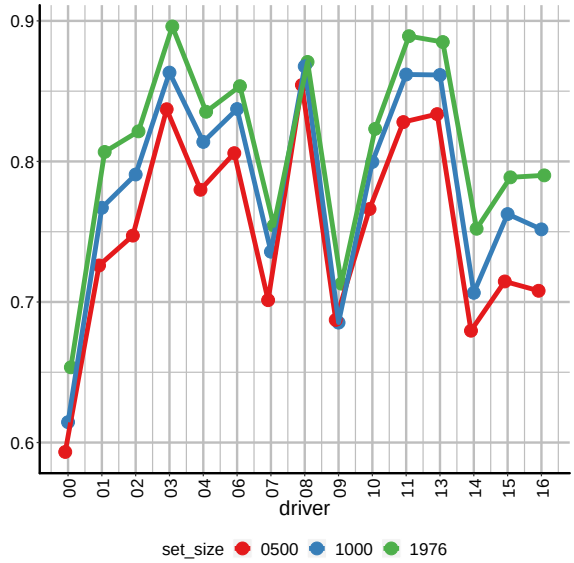
(c) Effect of data set size vs. matrix form.

,

Figure 8.18: Effects plots of the main effect of the data set size and interactions between the data set size and model and data set size and matrix form. On the y-axis is the AUC value, on the x-axis the (first) varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times$ SE. Results were obtained using the $n1976$-data set running elastic net models for all levels of α and both random forest options at a word count threshold of 11 for all matrix types. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

So it is clear that the performance increases as the size of the data set increases, both in general, when we look at the average over all models and matrix forms, as well as when we look at the models/matrix forms individually. Note though that the x-axis is non-linear, as 500, 1000 and 1976 are the ticks on the axis. There are some subtle differences between models though. For example, the Elastic net model with $\alpha = 0$, i.e. Ridge regression, performs the worst among all models, but the difference in AUC-value seems to be larger when the size of the data set is smaller. In general though, we see the same effect occur among all models and matrix forms. Also note that the difference in performance between 500 and 1000 seems to generally be the same as between 1000 and 1976. This is interesting, as in both cases it is roughly a doubling of the data set size. One would intuitively expect the curve of the AUC value would have a logarithmic form, but the first '$\times 2$' of the set size has seemingly the same effect as the second '$\times 2$'. This means that either the curve does not have a logarithmic form but a linear one, or, we are really only at the start of the curve of the logarithmic form. In both cases, this means that there is potential for our models to perform better, if more data was available. The 'promise of big data', more data leading to more accurate predictions, seems to hold here!

Now for the interaction between data set size and the individual drivers.



(a) Effects plot for the interaction between driver number and data set size.

0.	Shaving (general)
1.	During shaving – speed
2.	During shaving – comfort
3.	After shaving – irritation
4.	After shaving – closeness
5.	Cofo
6.	Sound
7.	Ease of use
8.	Cleaning
9.	Smart Clean
10.	Service & Delivery
11.	Accessories
12.	Features
13.	Charging & electronic power
14.	Design
15.	Value for money
16.	Issues & reliability

(b) Drivers corresponding to the driver numbers in the effect plots.

Figure 8.19: Effect plot of driver vs. data set size. On the y-axis is the AUC value, on the x-axis the (first) varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times \text{SE}$. Results were obtained using the $n1976$-data set running elastic net models for all levels of α and both random forest options at a word count threshold of 11 for all matrix types. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

What we see is that overall, the effect of an increase in the size of the data set is beneficial for all drivers. For some more than for others though. For driver 8. Cleaning the performance seems to be equal for all data set sizes. A possible explanation for this is found in the fact that our models use a certain set of words to determine whether driver 8 is discussed (or for positive and negative sentiment, driver 8 is discussed with a certain sentiment). The vocabulary used when talking about 'Cleaning' might be limited to such a small set of words, that these words are present in sufficient numbers when using a data set size of 500, and there is thus no further benefit to using larger data set sizes. Overall, this seems to be exceptional though. For all drivers except drivers 8 and 9 we see a clear benefit to the use of a larger data set size.

8.2. Results for the multinomial setting

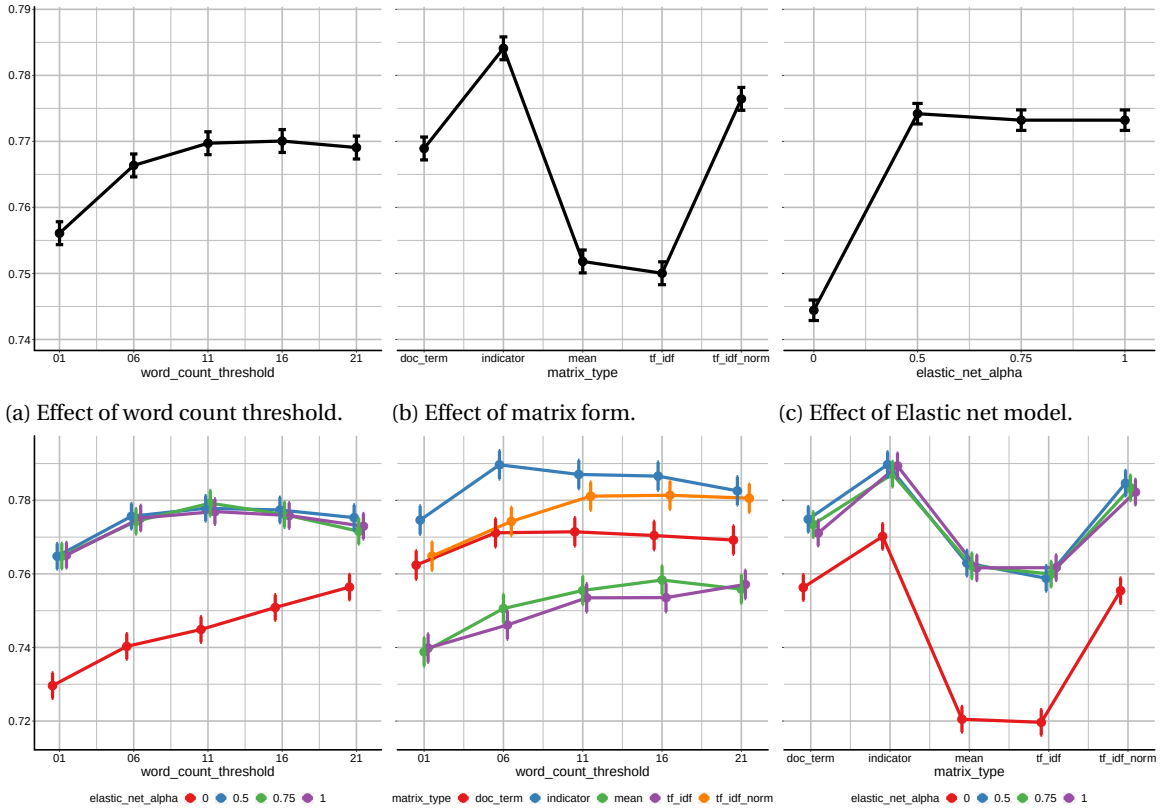
Now we come to the results for the multinomial setting. Again, we will first discuss the Elastic net models, then the random forest models and then compare them. We will then also show the performance on an individual driver level. Because the 3 types of sentiment are incorporated into 1 model in the multinomial setting, rather than 3 separate models in the binary setting, we cannot compare performance for separate sentiments here. Once again, we will first discuss results using the $n1976$-data set, and then using the $n775$-data set. Note that, different to the binary setting, we do not work with different types of sentiment in the multinomial setting. Thus, for the multinomial setting we produce 5 types of matrices × 5 word count threshold levels × 4 Elastic net models = 100 sets of results for 15 drivers.

There is another choice that has to be made for the multinomial setting though. The misclassification matrix A has to be chosen, in order to compute the AUC_μ, as introduced in Chapter 4. We need it both to evaluate our results and as the measure to train the Elastic net models on, as discussed in Chapter 5. We will choose the matrix A as

prediction(↓), true value (→)	P	N	NA
P	0	1	1
N	1	0	1
NA	1	1	0

Figure 8.20: Choice of matrix A

We will first extensively discuss all results obtained using this choice of A. Later in this chapter, we will discuss results obtained using a different choice of A will be discussed.



(a) Effect of word count threshold.

(b) Effect of matrix form.

(c) Effect of Elastic net model.

(d) Effect of word count threshold vs. Elastic net model.

(e) Effect of word count threshold vs. matrix form.

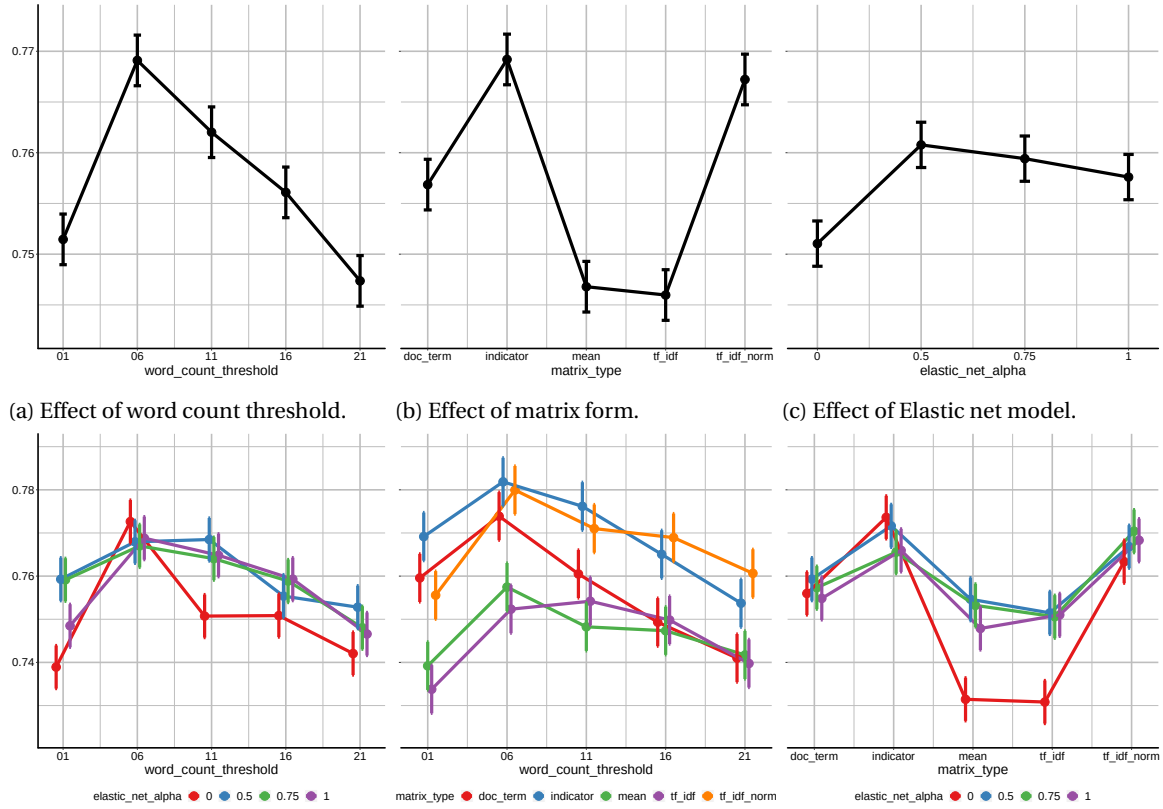
(f) Effect of matrix form vs. Elastic net model.

Figure 8.21: Main effects plots and interaction effects for the effects of the matrix form, word count threshold and Elastic net models. On the y-axis is the AUC_μ value, on the x-axis the (first) varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times \text{SE}$. Results were obtained using the $n1976$-data set. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

When it comes to the variability of the points, the points in Figures 8.21a and 8.21b are averages over $100 \times 15/5 = 300$ AUC estimates, the points in Figure 8.21c over $100 \times 15/4 = 375$ AUC estimates, the points in Figures 8.21d and 8.21f over $100 \times 15/(5 \times 4) = 75$ AUC estimates and the points in Figure 8.21e over $100 \times 15/(5 \times 5) = 60$ AUC estimates.

In Figure 8.21a, we see that the Elastic net models perform best at a word count threshold of 11 or 16, or maybe 21. This is contrary to the binary setting, where a word count threshold of 6 performed best (Figure 8.2a). Figure 8.21d shows that the effect occurs for all Elastic net models except when $\alpha = 0$ (Ridge regression). Furthermore, Figure 8.21e shows that for the indicator matrix form, the best performing among the matrix forms, a word count threshold of 6 performs best. It is for all others that a higher word count threshold gives the same or a higher average AUC value. Given the confidence intervals in Figure 8.21e, we have to be careful with his last statement though. For the binary setting, we see in Figure 8.3b performance of the indicator matrix form is also more or less the same at word count thresholds 6, 11 and 16.

Furthermore, what stands out from Figures 8.21e, 8.21f and 8.21c in general is the severe under-performance of Ridge regression in the multinomial setting. In the binary setting we also saw Ridge regression perform worse than the other Elastic net models, but less so. From these same figures, we see that the other 3 Elastic net models perform equally good at all word count threshold levels and for all matrix forms. This effect was there in the binary setting as well (Figures 8.3a and 8.3c), but less profound.



(a) Effect of word count threshold.

(b) Effect of matrix form.

(c) Effect of Elastic net model.

(d) Effect of word count threshold vs. Elastic net model.

(e) Effect of word count threshold vs. matrix form.

(f) Effect of matrix form vs. Elastic net model.

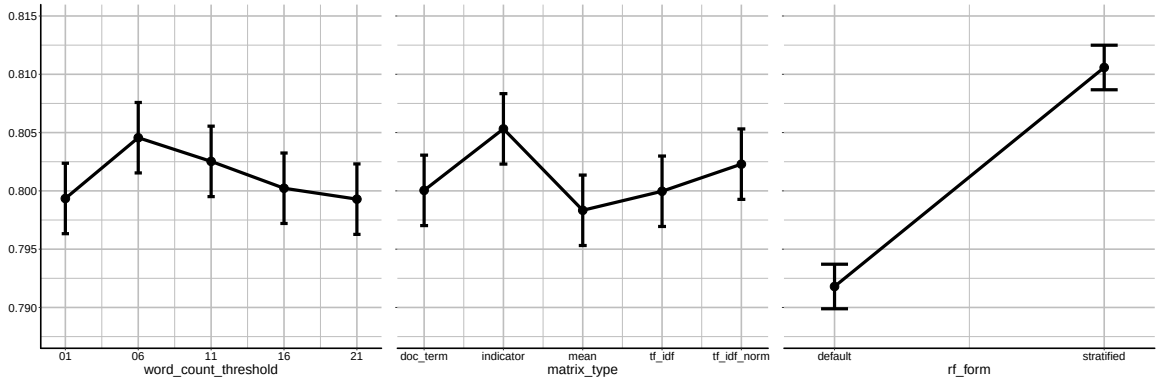
Figure 8.22: Main effects plots and interaction effects for the effects of the matrix form, word count threshold and Elastic net models. On the y-axis is the AUC_μ value, on the x-axis the (first) varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times SE$. Results were obtained using the $n775$-data set. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

When it comes to the variability of the points, the points in Figures 8.22a and 8.22b are averages over $100 \times 14/5 = 280$ AUC estimates, the points in Figure 8.22c over $100 \times 14/4 = 350$ AUC estimates, the points in Figures 8.22d and 8.22f over $100 \times 14/(5 \times 2) = 140$ AUC estimates and the points in Figure 8.22e over $100 \times 14/(5 \times 5) = 56$ AUC estimates.

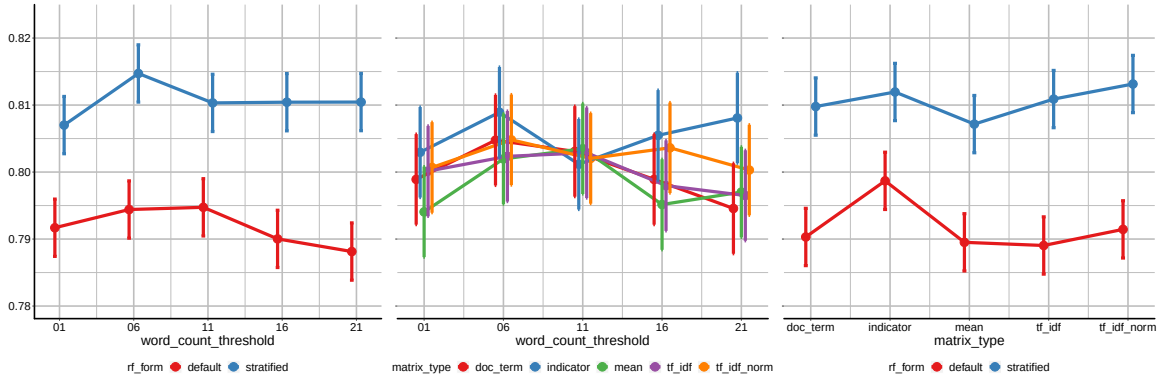
To a certain extent we see the same effects occur here, in the multinomial setting, as we saw in the binary setting when applying Elastic net models for the $n775$-data set (Figures 8.4 and 8.5). What stands out is the good performance of the normalized tf-idf matrix form as seen in Figure 8.22b, which we did not see in Figure 8.4a. Also when looking at the interaction plots in Figures 8.22e and 8.22f, we see this good performance of the normalized tf-idf matrix form occur at all word count threshold levels and for all Elastic net models except $\alpha = 0$ (=Ridge regression).

To an extent, we also saw this effect happen when applying Elastic net models in the multinomial setting for the $n1976$-data set, in Figure 8.21. The performance of the normalized tf-idf matrix form equaled that of the indicator matrix form at best, but given that the indicator matrix form was the best everywhere, this is already an achievement in itself. It seems that the Elastic net models gain more value from using the normalized tf-idf values as input in the multinomial setting than they do when using these (exact same) values in the binary setting.

8.2.1. Random forest models



(a) Effect of word count threshold. (b) Effect of matrix form. (c) Effect of Elastic net model.



(d) Effect of word count threshold vs. Elastic net model. (e) Effect of word count threshold vs. matrix form. (f) Effect of matrix form vs. Elastic net model.

Figure 8.23: Main effects plots and interaction effects for the effects of the matrix form, word count threshold and Elastic net models. On the y-axis is the AUC_μ value, on the x-axis the (first) varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times SE$. Results were obtained using the $n1976$-data set. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

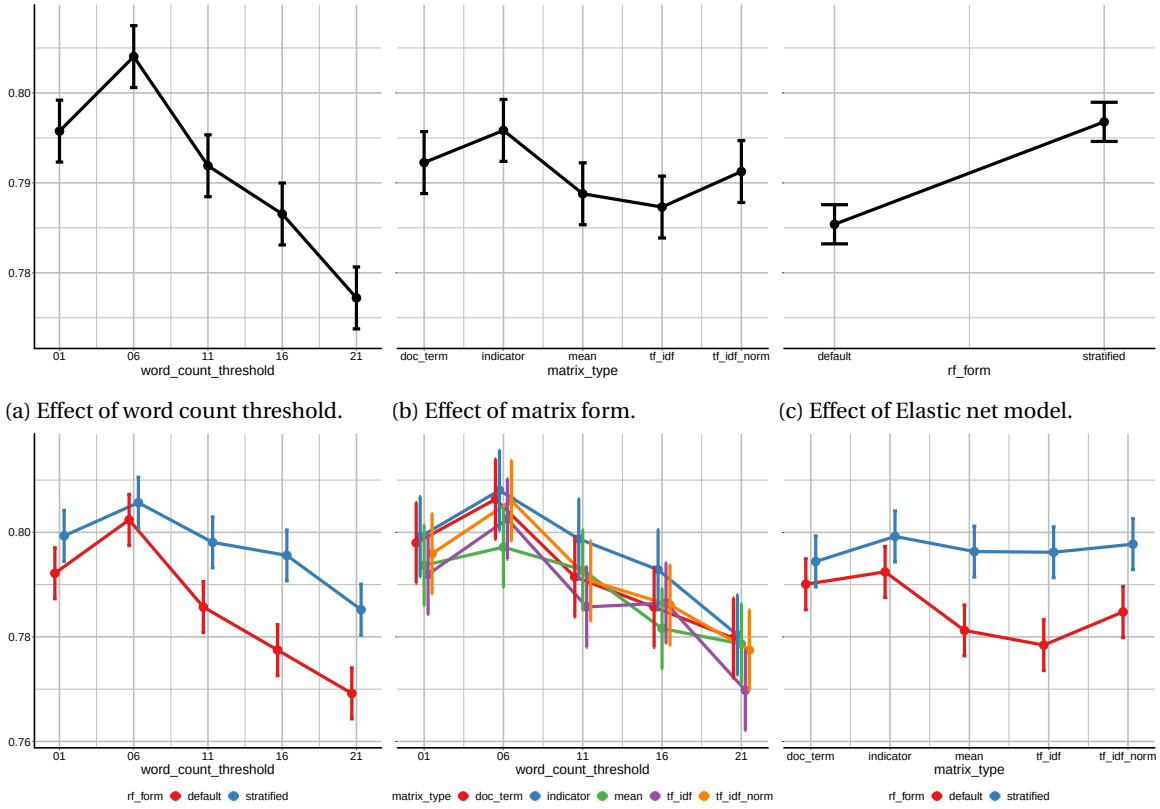
When it comes to the variability of the points, the points in Figures 8.23a and 8.23b are averages over $50 \times 15/5 = 150$ AUC estimates, the points in Figure 8.23c over $50 \times 15/2 = 375$ AUC estimates, the points in Figures 8.23d and 8.23f over $100 \times 15/(5 \times 2) = 150$ AUC estimates and the points in Figure 8.24e over $100 \times 15/(5 \times 5) = 60$ AUC estimates. Furthermore, the confidence intervals here may at first glance seem larger than the ones we saw when looking at the results for the Elastic net models, but one should note that the scale on the y-axis has also changed, so they are quite comparable.

For these results, the ANOVA results for the interaction effects in Figures 8.23d, 8.23e and 8.23f are particularly interesting:

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
matrix_type:rf_form	4	0.00	0.00	1.33	0.2591
matrix_type:word_count_threshold	16	0.00	0.00	0.62	0.8712
rf_form:word_count_threshold	4	0.00	0.00	1.05	0.3825

All three interaction effects that we plotted have a p-value > 0.05, and are thus considered insignificant. Considering the confidence intervals around the plotted points, it is also hard to observe more from the interaction plots than that overall the stratified random forest model still performs better than the default option, as was also the case in the binary setting (Figures 8.7a and 8.7c). Furthermore, similar as to what we saw for the

Elastic net models, we see in Figure 8.23b that in the multinomial setting the normalized tf-idf matrix form can compete with the indicator matrix form.



(a) Effect of word count threshold.

(b) Effect of matrix form.

(c) Effect of Elastic net model.

(d) Effect of word count threshold vs. Elastic net model.

(e) Effect of word count threshold vs. matrix form.

(f) Effect of matrix form vs. Elastic net model.

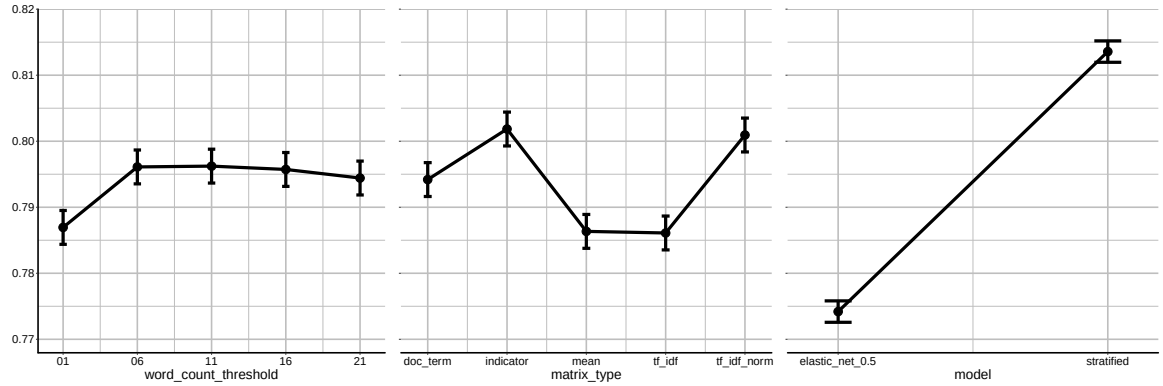
Figure 8.24: Main effects plots and interaction effects for the effects of the matrix form, word count threshold and Elastic net models. On the y-axis is the AUC_μ value, on the x-axis the (first) varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times \text{SE}$. Results were obtained using the n775-data set. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

When it comes to the variability of the points, the points in Figures 8.24a and 8.24b are averages over $50 \times 14/5 = 140$ AUC estimates, the points in Figure 8.24c over $50 \times 14/2 = 350$ AUC estimates, the points in Figures 8.24d and 8.24f over $100 \times 14/(5 \times 2) = 140$ AUC estimates and the points in Figure 8.24e over $100 \times 14/(5 \times 5) = 56$ AUC estimates.

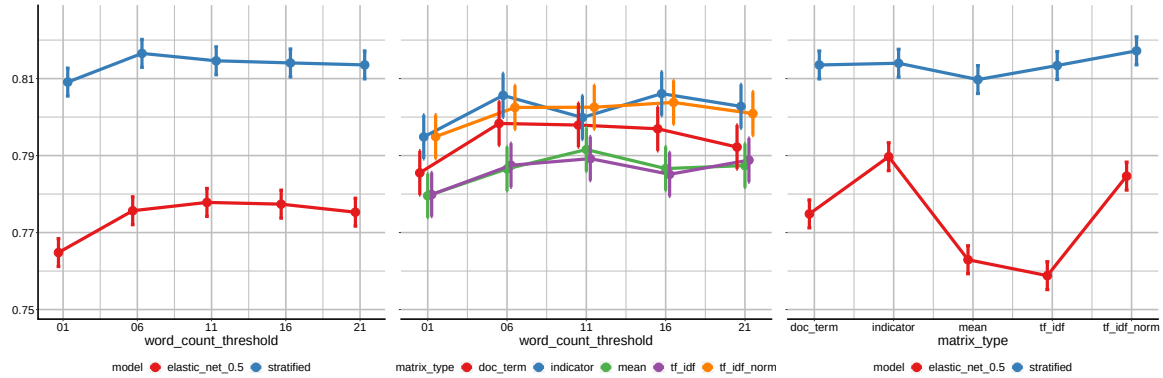
When it comes to the ANOVA values, the interaction between word count threshold and matrix from from Figure 8.24e is considered insignificant, with a p-value of 0.96. Unlike to the results for the n1976-data set as shown in Figure 8.23, the other two interactions displayed here in Figures 8.24d and 8.24f can be considered significant, although it is rather close with p-values of 0.04 and 0.01 respectively. The results are very similar to what we saw happen in the binary setting when using random forests for the n775-data set in Figures 8.8c and 8.9. The good performance of the the stratified option random forest at a word count threshold of 6 is odd, but we saw the same thing happen in the binary setting when using the n775-data set. In conclusion, when comparing the results of using the random forests in the multinomial setting for the n1976-data set (Figure 8.23) and the n775-data set (Figure 8.24), we see subtle differences in the effect of the word count threshold and the matrix form. These differences are the same as we saw when comparing the use of random forests for these two data sets in the binary setting though.

What stands out is that the normalized tf-idf matrix seemingly performs better than the indicator matrix form, or at least as good. For the n1976-data performance seemed about equal, and also when using the n775-data set the difference is too small to say it with confidence, but, if were asked to choose a matrix form when applying a (stratified) random forest on an arbitrary data set, we would choose the normalized tf-idf matrix form over the indicator matrix form, based on these results.

8.2.2. Comparison random forest and Elastic net models



(a) Effect of word count threshold. (b) Effect of matrix form. (c) Effect of model.

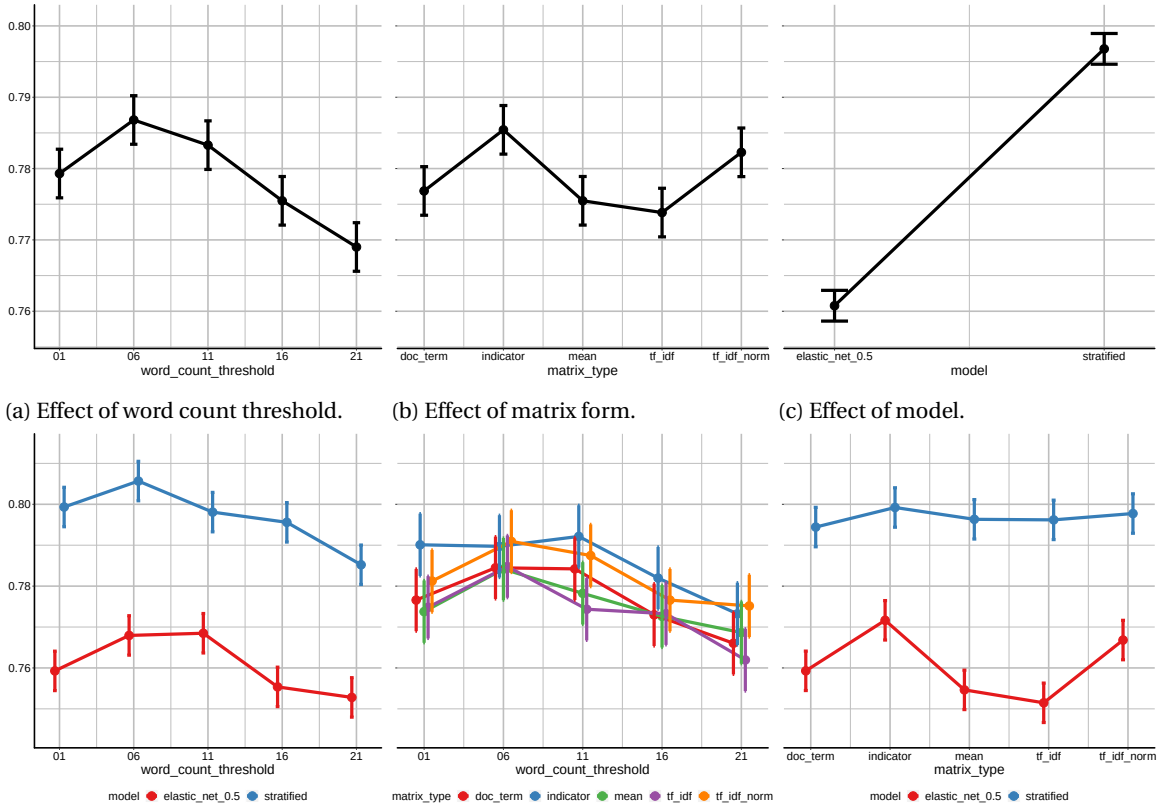


(d) Effect of word count threshold vs. model. (e) Effect of word count threshold vs. matrix form. (f) Effect of matrix form vs. model.

Figure 8.25: Main effects plots and interaction effects for the effects of the matrix form, word count threshold and Elastic net models. On the y-axis is the AUC_μ value, on the x-axis the (first) varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times SE$. Results were obtained using the $n1976$-data set. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

When it comes to the variability of the points, the points in Figures 8.25a and 8.25b are averages over $50 \times 15/5 = 150$ AUC estimates, the points in Figure 8.25c over $50 \times 15/2 = 375$ AUC estimates, the points in Figures 8.25d and 8.25f over $100 \times 15/(5 \times 2) = 150$ AUC estimates and the points in Figure 8.25e over $100 \times 15/(5 \times 5) = 60$ AUC estimates.

The interaction effects of the word count threshold and model (Figure 8.25d) and word count threshold and matrix form (Figure 8.25e) are considered insignificant, with p-values of 0.19 and 0.91 respectively. Comparing the results in Figure 8.25 with their equivalent in the binary setting (Figures 8.10 and 8.11), we see no large differences. Of interest to us is the slightly better performance of the normalized tf-idf matrix form here. Furthermore, it is important to note that, also in the multinomial setting, the stratified random forest model performs better than the Elastic net model with $\alpha = 0.5$.



(a) Effect of word count threshold.

(b) Effect of matrix form.

(c) Effect of model.

(d) Effect of word count threshold vs. model.

(e) Effect of word count threshold vs. matrix form.

(f) Effect of matrix form vs. model.

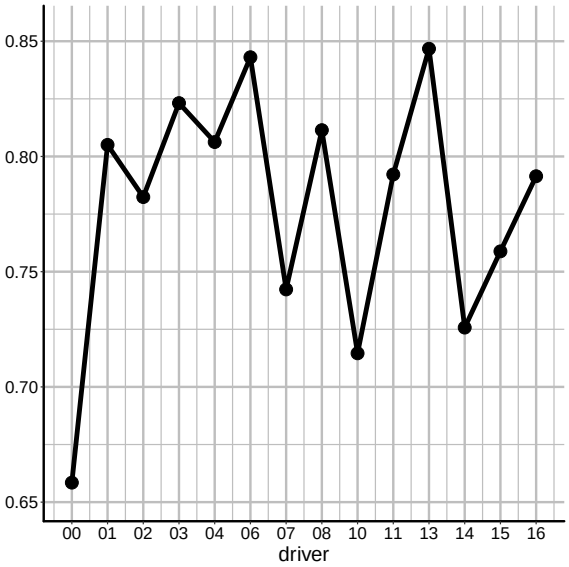
Figure 8.26: Main effects plots and interaction effects for the effects of the matrix form, word count threshold and Elastic net models. On the y-axis is the AUC_μ value, on the x-axis the (first) varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times \text{SE}$. Results were obtained using the n775-data set. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

When it comes to the variability of the points, the points in Figures 8.26a and 8.26b are averages over $50 \times 14/5 = 140$ AUC estimates, the points in Figure 8.26c over $50 \times 14/2 = 350$ AUC estimates, the points in Figures 8.26d and 8.26f over $100 \times 14/(5 \times 2) = 140$ AUC estimates and the points in Figure 8.26e over $100 \times 14/(5 \times 5) = 56$ AUC estimates.

Once again, the ANOVA results reveal that the interaction effects of the word count threshold and model (Figure 8.26d) and word count threshold and matrix form (Figure 8.26e) are considered insignificant, with p-values of 0.11 and 0.94 respectively. When comparing the results in Figure 8.26 with those obtained using the n775-data set in the binary setting in Figures 8.12 and 8.13, we see that the effect of the word count threshold seems slightly different here. As the word count threshold increases, the average performance of models using word count threshold greater than 6 seems to reduce further here, in the multinomial setting, than it does in the binary setting. Given the confidence intervals around the means, this judgement has to be taken with a grain of salt though. Using these results and those from Figure 8.25, the conclusion from the comparison between the (stratified) random forest model and the Elastic net model with $\alpha = 0.5$, is that the performance of the random forest is higher than that of the Elastic net model. This makes it our preferred choice of model.

8.2.3. Driver results

As for the binary setting, we will look at the performance of the models for the individual drivers. In the binary setting, we also looked at models for different types of sentiment. As we no longer have different models for different sentiments in the multinomial setting, but rather have one model that incorporates them, we only look at the drivers. We start with the results for the results for the n1976-data set.



(a) Effect of driver number.

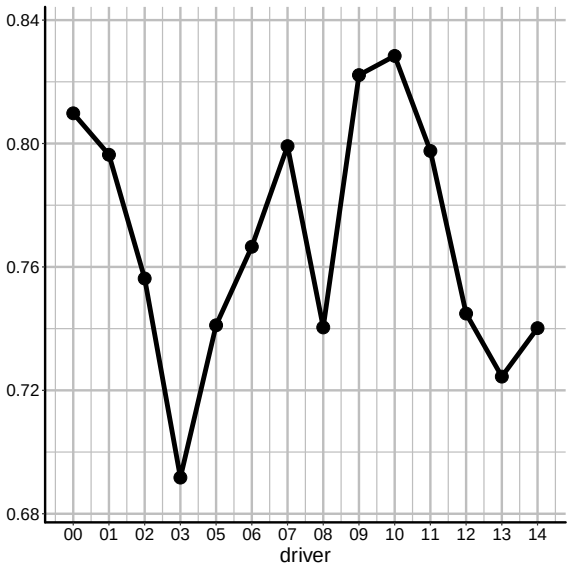
0.	Shaving (general)
1.	During shaving – speed
2.	During shaving – comfort
3.	After shaving – irritation
4.	After shaving – closeness
5.	Cofo
6.	Sound
7.	Ease of use
8.	Cleaning
9.	Smart Clean
10.	Service & Delivery
11.	Accessories
12.	Features
13.	Charging & electronic power
14.	Design
15.	Value for money
16.	Issues & reliability

(b) Drivers corresponding to the driver numbers in the effect plots.

Figure 8.27: Effects plot for the effect of driver number. On the y-axis is the AUC_μ value, on the x-axis the varying parameter for the driver number. A list of drivers corresponding to the driver numbers is provided. Results were obtained using the n1976-data set running elastic net models for all levels of α and both random forest options at all word count thresholds for all matrix types. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

When we compare these results with the results we found in the binary setting (Figure 8.14), we find that they are very comparable. Generally, it are the same drivers that perform well in the multinomial setting that do well in the binary setting. The relative ranking of the performance of the drivers has changed a bit though. For example, driver 06. - Sound performs better than driver 03. After shaving - irritation according to the results in Figure 8.27. In the binary setting, we saw in Figure 8.14 the opposite: 06. - Sound performed worse than 03. After shaving - irritation, also for different types of sentiment. Of course it is not strange that this occurs to some extent. The multinomial models are different from the binary models, and the we have a different value that is measured. Nevertheless, the fact that when we zoom out and look at the grand scheme of driver performance, we see the same drivers perform well as that we see in the binary setting, gives us more confidence that the AUC_μ is well chosen as a measure for the multinomial setting. The reasons why our models perform better for some drivers than for others, is therefore the same hypothesis that we formulated in the binary setting. It seems that for some drivers it is true that the number of reviews discussing them in the training data is relevant, while for others it seems that they are inherently easier to predict.

In Figure 8.28 we have also displayed the results for the drivers in the n775-data set. Generally, the same observation as for the n1976-data set holds. When comparing with the binary setting (Figure 8.17) we generally see the same drivers perform well and the same drivers perform poorly. There are differences, but the general trend has remained the same.



(a) Effect of driver number.

1.	Accessories
2.	Charging & Electronic power
3.	Design
4.	Features
5.	Issues & Reliability
6.	Service & Delivery
7.	Value for money
8.	cleaning
9.	ease of use
10.	shaving closeness
11.	shaving comfort
12.	shaving irritation
13.	shaving rest
14.	shaving speed
15.	smart clean

(b) Drivers corresponding to the driver numbers in the effect plots.

Figure 8.28: Effects plot for the effect of driver number. On the y-axis is the AUC_{μ} value, on the x-axis the varying parameter for the driver number. A list of drivers corresponding to the driver numbers is provided. Results were obtained using the n775-data set running elastic net models for all levels of α and both random forest options at all word count thresholds for all matrix types. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

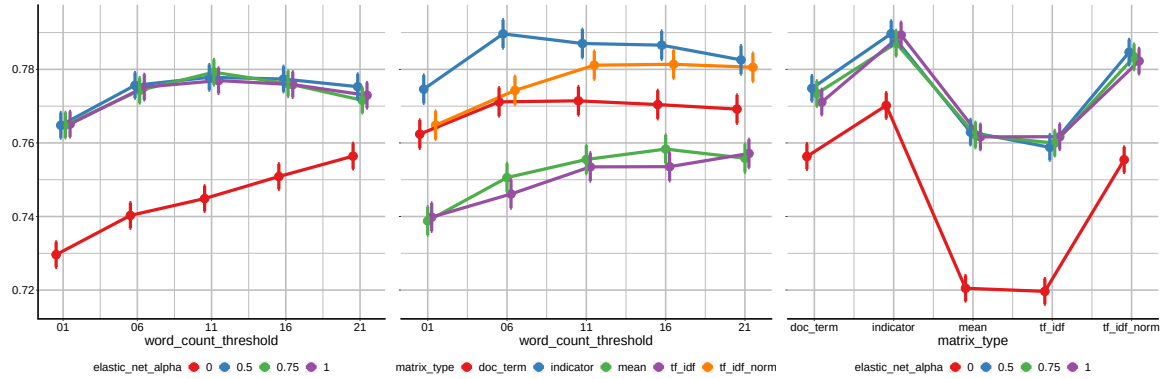
8.3. A different misclassification matrix

Now, we also generated results using a different choice of misclassification matrix A. Figure B.1 shows how A was chosen. This choice of A was used because it increased the cost of misclassifying a P/N as NA or vice versa, which is considered the more grievous error.

prediction(↓), true value (→)	P	N	NA
P	0	3/5	6/5
N	3/5	0	6/5
NA	6/5	6/5	0

Figure 8.29: Choice of matrix A

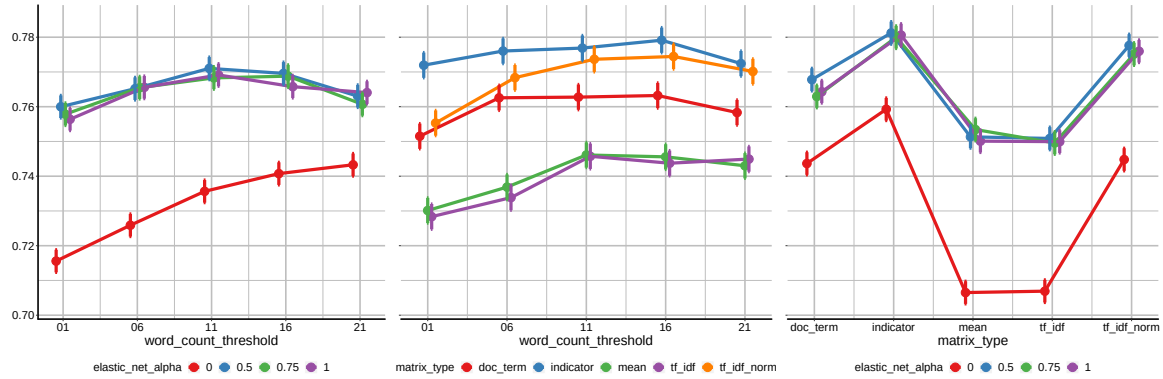
The results of using this misclassifcation matrix are rather disappointing though. They are very similar to the results using a matrix A chosen such that all misclassifcation cost are equal. As an example, look at the interaction plots for the Elastic net models using the n1976-data set.



(a) Effect of word count threshold vs. Elastic net model.

(b) Effect of word count threshold vs. matrix form.

(c) Effect of matrix form vs. Elastic net model.



(d) Effect of word count threshold vs. Elastic net model.

(e) Effect of word count threshold vs. matrix form.

(f) Effect of matrix form vs. Elastic net model.

Figure 8.30: Interaction effects for the effects of the matrix form, word count threshold and Elastic net models. Plotted for models using two choices of misclassification matrix A. Results for A as in Figure 8.20 on top, for A as in Figure 8.29 on the bottom. On the y-axis is the AUC_μ value, on the x-axis the (first) varying parameter for its respective effects plot. The displayed confidence intervals are calculated as $1.96 \times \text{SE}$. Results were obtained using the n1976-data set. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

As one can see, the results are very similar, if not the same considering the error bars. The results of using the matrix A as in Figure 8.29 are therefore not discussed in this thesis. They are are printed in Appendix B.

9

Application

We have discussed the more technical results in Chapter 8, where we discussed how different choices of parameters influenced the performance of our models. In this chapter, we will use these models to zoom in on the performance of some of our models in the binary setting. We will first look at how the driver scores for a review can be predicted using our models. Then we will take a closer look at how the coefficients in the Elastic net model influence the prediction of a review and show how these can be compared to important variables in a random forest model.

9.1. Example of review annotation

Say that we have trained a model, and use it to predict a new review. An example of a new review is:

"I have been extremely pleased with this shaver. It shaves close, is fast, and is relatively quiet compared with the previous electric shaver that I had. Would highly recommend this product."

Figure 9.1: Example of a review in the n1976-data set.

This particular review is part of the n1976-data set. It is annotated for the 17 drivers in this data set as

Shaving (general)	During shaving – speed	During shaving – comfort	After shaving – irritation	After shaving – closeness	Cofo
2	2	NA	NA	2	NA
Sound	Ease of use	cleaning	Smart Clean	Service & Delivery	Accessories
1	NA	NA	NA	NA	NA
Features	Charging & electronic power	Design	Value for money	Issues & reliability	
NA	NA	NA	NA	NA	

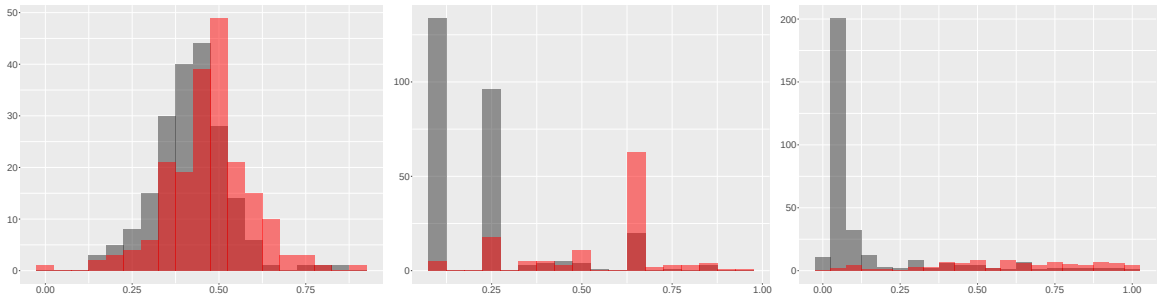
Figure 9.2: Annotated driver scores for the review in Figure 9.1

Say that we have build a model as we have done in this thesis for prediction of present sentiment, i.e., whether a review discusses a given driver or not, and the review from Figure 9.1 was not used to train the model. When we give the review from Figure 9.1 as input for the model, we obtain a set of estimated probabilities, $\widehat{p}_{i,1},\ldots,\widehat{p}_{i,17}$, i being the index of the review. In the case of an Elastic net, these estimated probabilities are found using the estimated coefficients (from the logistic regression) and in the case of the random forest it is the fraction of votes from the decision trees.

Shaving (general)	During shaving – speed	During shaving – comfort	After shaving – irritation	After shaving – closeness	Cofo
0.4829714	0.19303905	0.037321740	0.03907520	0.6510615	NA
Sound	Ease of use	cleaning	Smart Clean	Service & Delivery	Accessories
0.42506108	0.03540277	0.02524224	0.005827753	0.014796885	0.02777489
Features	Charging & electronic power	Design	Value for money	Issues & reliability	
NA	0.038897240	0.03446941	0.1129507	0.04584707	

Figure 9.3: Annotated driver scores for the review in Figure 9.1

This particular example was obtained using an Elastic net model with $\alpha = 0.5$ trained on 80% of the n1976-data set with a word count threshold of 6 and and indicator matrix form. The review from Figure 9.1 was part of the 20% of the n1976-data set not used, the test set. Probability estimates were not noted in Figure 9.3 for 2 drivers, because, as discussed before in this thesis, there are not enough reviews discussing these drivers to build an effective model. Now we still only have probability estimates though. We will have to set a threshold value t on these probabilities to get the estimates for the driver scores. How will we choose these thresholds though? With the goal of a (semi-)automated annotation tool in our mind and armed with the fact that there generally is a large imbalance between the number of 1s and 0s in our data set, we aim to set threshold such that we will at least predict a certain fraction of those 1s correctly. This means that we aim for a certain sensitivity (= true positives / (true positives + false negatives). How do we set this threshold then? First, it is important to understand that we will need to set a different threshold value for every driver. Let us illustrate this with a couple of histograms of the estimated probabilities for the 20% test set of which the review from Figure 9.1 was a part.



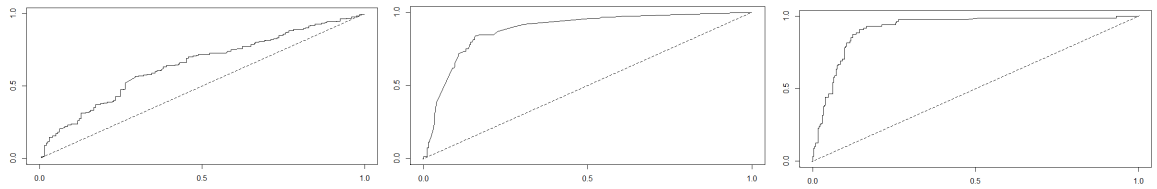
(a) 0.Shaving (general) (b) 4.After shaving – closeness (c) 13.Charging & electronic power

Figure 9.4: Overlaying histograms of the estimated probabilities for reviews annotated as a 1 (red) or 0 (gray) in the test set for drivers 0.Shaving (general), 4.After shaving – closeness and 13.Charging & electronic power in the data set. On the x-axis is the estimated probability between 0 and 1.

Driver 0. Shaving (general) is the odd one out here, as was noted earlier in this thesis. There are many reviews for which it is annotated as non-NA, and is annotated as representing the general sentiment in a review. It is by far not always annotated in the data set though, so when predicting present-sentiment as we are doing now, there are still plenty of 0s and not only 1s to predict. We also see that there is a lot of overlap between the estimated probabilities for the 1s and the 0s for this driver. For 4.After shaving – closeness and 13.Charging & electronic power this is much less so the case. Reviews corresponding to a 1, also seem to generally have a higher estimated probability. This makes us think of how the AUC can be interpreted as being the probability that a 1 is ranked higher than a 0, as was shown in Chapter 4. When computing the AUC values using the results from this test set, we find an AUC value of 0.65 for 0. Shaving (general), 0.88 for 4.After shaving – closeness and 0.92 for 13.Charging & electronic power.

We move back to searching the threshold value t to go from the estimated probabilities to the actual predictions for the driver scores. In Figure 9.4 it becomes clear that setting the same threshold value t for all drivers, leads to large differences in the accuracy of these predictions. Specifically, we are interested in the relation

between the sensitivity and specificity at a certain threshold. What one desires will depend on the application, but a logical choice in our opinion would be to aim for at least a certain specificity, so that you detect at least a certain number of 1s. Plotting ROC curves for the drivers can help to see what level of sensitivity can be achieved, without the specificity being too low. We plot the ROC curves corresponding to the results in Figure 9.4.



(a) 0.Shaving (general) (b) 4.After shaving – closeness (c) 13.Charging & electronic power

Figure 9.5: ROC curves of the estimated probabilities for reviews annotated as a 1 (red) or 0 (gray) in the test set for drivers 0.Shaving (general), 4.After shaving – closeness and 13.Charging & electronic power in the data set. On the x-axis is 1−specificity, on the y-axis the sensitivity..

Doing this for all drivers, we set thresholds for different levels of specificity and sensitivity.

Shaving (general)	During shaving – speed	During shaving – comfort	After shaving – irritation	After shaving – closeness	Cofo
$t = 0.442$ sensitivity= 0.65 specificity= 0.57	$t = 0.176$ sensitivity= 0.80 specificity= 0.71	$t = 0.045$ sensitivity= 0.85 specificity= 0.75	$t = 0.059$ sensitivity= 0.85 specificity= 0.82	$t = 0.230$ sensitivity= 0.85 specificity= 0.77	NA
Sound	Ease of use	cleaning	Smart Clean	Service & Delivery	Accessories
$t = 0.039$ sensitivity= 0.90 specificity= 0.97	$t = 0.048$ sensitivity= 0.75 specificity= 0.74	$t = 0.223$ sensitivity= 0.90 specificity= 0.89	$t = 0.008$ sensitivity= 0.95 specificity= 0.76	$t = 0.101$ sensitivity= 0.65 specificity= 0.95	$t = 0.038$ sensitivity= 0.85 specificity= 0.86
Features	Charging & electronic power	Design	Value for money	Issues & reliability	
NA	$t = 0.282$ sensitivity= 0.90 specificity= 0.86	$t = 0.036$ sensitivity= 0.75 specificity= 0.68	$t = 0.134$ sensitivity= 0.78 specificity= 0.82	$t = 0.065$ sensitivity= 0.72 specificity= 0.68	

Figure 9.6: Annotated driver scores for the review in Figure 9.1

If we then apply these threshold values to the review we introduced in Figure 9.1, we get our predictions for the driver scores, $\widehat{d}'_i$, and can compare these with true values d'_i.

Shaving (general)	During shaving – speed	During shaving – comfort	After shaving – irritation	After shaving – closeness	Cofo
$\widehat{d}'_1 = 1$ $d'_1 = 1$	$\widehat{d}'_2 = 1$ $d'_2 = 1$	$\widehat{d}'_3 = 0$ $d'_3 = 0$	$\widehat{d}'_4 = 0$ $d'_4 = 0$	$\widehat{d}'_5 = 1$ $d'_5 = 1$	NA
Sound	Ease of use	cleaning	Smart Clean	Service & Delivery	Accessories
$\widehat{d}'_7 = 1$ $d'_7 = 1$	$\widehat{d}'_8 = 0$ $d'_8 = 0$	$\widehat{d}'_9 = 0$ $d'_9 = 0$	$\widehat{d}'_{10} = 0$ $d'_{10} = 0$	$\widehat{d}'_{11} = 0$ $d'_{11} = 0$	$\widehat{d}'_{12} = 0$ $d'_{12} = 0$
Features	Charging & electronic power	Design	Value for money	Issues & reliability	
NA	$\widehat{d}'_{14} = 0$ $d'_{14} = 0$	$\widehat{d}'_{15} = 0$ $d'_{15} = 0$	$\widehat{d}'_{16} = 0$ $d'_{16} = 0$	$\widehat{d}'_{17} = 0$ $d'_{17} = 0$	

Figure 9.7: Predicted and actual (coarsened) driver scores for the review in Figure 9.1

So we see that for this particular review, we have reached a particularly good result with our chosen thresholds, as the predictions for all drivers are correct. It is not always the case that our predictions turn out so well though.

9.2. Selected words and reviews

Let us take a closer look at one driver and a few reviews. We take driver 12. 'shaving irritation' from the $n775$-data set and train an Elastic net model with $\alpha = 1$ (i.e., a LASSO) on 80% of the $n1976$-data set with a word count threshold of 11 and the indicator matrix form. Let us first look at for which variables the coefficients are estimated as non-zero.

variable	estimated coefficient
neck	0.0652
irrit	3.31
skin	1.10
star	0.120
face	0.838
pai	0.890
recharg	0.747
left	0.907
rash	3.08
burn	1.85
forev	0.380
self	0.00211

Figure 9.8: Estimated non-zero coefficients for driver 12. 'shaving irritation'.

We see that the larger coefficients, that stand out among the others, are 'irrit', 'skin' and 'rash'; words that we, as regular who happen to speak English, associate with skin irritation after shaving. Let us now apply the model on the 20% of the reviews that we have left. Let us look at a few reviews that receive a relatively high estimated probability $\widehat{p}$. Because some reviews are very long, sometimes only relevant excerpts are printed.

"...It was quick but not a great shave so as I got older, switched to blades. Seeing the ads and these reviews I thought to try the 'new' shaving experience and went with the top of the line model.Well, I had to press hard and go over areas so many times that every little raised imperfection in the skin was irritated.The trouble spots on the neck just did not get a close shave and I found myself going over them with a blade any way. I consistently got a rash and burn. ..."

(a) Review for which $\widehat{p} = 0.99$

"I have never got on with electric shavers, they have always irritated my skin, so having filled out a survey saying I don't use an electric shaver I was surprised to be selected for the trial and I was sent this product for free. I have used other electric shavers and my neck and jaw line turn into a red rash almost immediately, I have been using the Braun Series 9 9296cc shaver for a week now and my skin is great. Now the razor itself is stylish and looks great, it's easy to use and it's 10-D Flex head that gently adapts to your skin contours ..."

(b) Review for which $\widehat{p} = 0.99$

"...doesn't shave me well. It leaves random (and visibly noticeable) patches of stubble on my face and random hairs (both stubble length and long). The only way to get rid of these stubble patches is to literally shave in every direction whilst pushing super hard onto my skin. Because of this my skin got really red and irritated everywhere and left me not wanting to shave ..."

(c) Review for which $\widehat{p} = 0.96$

Figure 9.9: Reviews extracted from the 20% subset of the reviews.

And indeed we see in the excerpts printed in Figure 9.9 that all discuss skin irritation, and use the relevant

coefficients, hence why they are assigned a rather high probability. What is odd though is that, where the reviews of which excerpts are shown in Figures 9.9a and 9.9c are annotated in the data as discussing driver 12. 'shaving - irritation', the review of which an excerpt is shown in Figure 9.9b is not: there is an NA filled in for its driver score. This is odd. Both the model and anyone reading the review would say it discusses skin irritation, but in the actual data it is not annotated as such. We can only conclude that there are sometimes mistakes in the data set, which can of course happen when manually annotating 775 of these reviews.

Two more examples of reviews and their estimated probabilities are given in Figure 9.10.

"... Thankfully this model was able to use all the parts from my old one. Pros: - The least irritating electric shave I've experienced over the last 20+ years ..."

(a) Review for which $\hat{p} = 0.73$

"I was fed up of my hubby looking like he had just had a fight with the cat, with bits of tissue sticking to his neck where he had nicked himself with the wet razor. He wasn't convinced that anything could be better than a wet shave ..."

(b) Review for which $\hat{p} = 0.12$

Figure 9.10: Reviews extracted from the 20% subset of the reviews.

For the review in Figure 9.10a, we see that the review uses the word 'irrititating', but not in the context of irritated skin. Hence, the review is not about shaving irritation, and is also not annotated as such in the data. Since the stem 'irrit' is one of the coefficients of our model, this causes $\hat{p}$ to be relatively high.

The review in Figure 9.10b is about skin irritation, and is also annotated as such. However the words used to describe shaving irritation do not correspond to the coefficients of our model, hence the low value for $\hat{p}$. The reviews on which the model was trained may not have used language such as 'bits of tissue' when discussing irritation, and therefore the value of $\hat{p}$ is rather low for this review. This does not mean that we cannot predict it correctly. With a low enough threshold value t, this review can still be predicted correctly. This would also mean that one has to accept a lower level of specificity though.

Let us take a look at another driver. We take driver 9. 'ease of use'. Where our model for 12. 'shaving irritation' had a relatively high AUC, the model for driver 9. 'ease of use' has a rather low one (see Figure 8.17 in Chapter 8). The coefficients for this model are:

variable	estimated coefficient
hold	1.10
on	0.252
place	0.197
love	0.127
work	-0.0223
come	0.250
awkward	1.25
lubric	0.414
heavi	1.05
light	0.303
button	1.27
handl	0.773
easi	1.52
sideburn	1.01
nose	0.387
grip	0.837
sore	0.507

Figure 9.11: Estimated non-zero coefficients for driver 9. 'ease of use'.

We see that words that stand out and receive a relatively large coefficient are 'hold', 'easi', 'sideburn', 'awkward'

and 'grip'. Not all these are words that we necessarily naturally associate with 'ease of use'. Let us take a look at the reviews that received the highest estimated probabilities. Note that while these probabilities might not seem high, the distribution of the estimated probabilities is for this driver such that these 3 are the highest in our 20% subset of the data.

"very nice and smooth. I love the way t holds and it has great battery life. The only thing I didn't like was that there was no room in the case for the beard trimmer which I thought was brainless. Easy to clean. The shave could . . . "

(a) Review for which $\widehat{p} = 0.52$

"Works the best of all the electric razors I have ever had. Close shave, long lasting battery, easy to hold with lots of shaving choices. Now for the real important info, my wife likes to use it on her legs. Which means its gentle enough for her and that saves me money on buying a separate razor for her. If I had to pick . . . "

(b) Review for which $\widehat{p} = 0.48$

". . . returning the razor if you are not satisfied. I like the Smart Clean system. It works well the but the on button is sometimes sensitive to touch if you are holding the unit when you insert the razor. Good product for the money. As far as some commenting about the shaver not having a pop up trimmer that is not a big deal. It makes this shaver lighter in weight when shaving."

(c) Review for which $\widehat{p} = 0.48$

Figure 9.12: Reviews extracted from the 20% subset of the reviews.

We observe that the coefficients of the model are used in these reviews, and therefore our model gives them a high probability. The first two reviews, in Figures 9.12a and 9.12b, are also annotated as discussing driver 9. 'ease of use'. The third one, in Figure 9.12c, is not. What really is 'ease of use' though? The first two reviews both mention that the shaver is 'easy to clean', the third one mentions that the reviewer likes 'the Smart Clean system'. So does 'easy to clean' mean that the shaver is easy to use? Another driver in the n775-data set is 8. 'Cleaning'. Should these reviews be annotated for either driver 8 or driver 9, or both of them? Another review annotated in the data as discussing driver 9. 'ease of use' is given in Figure 9.13.

"Great powerful, lightweight shaver and I am delighted with its performance! I've not felt the need or desire to use this in its wet format although I'm sure it work just as well wet!!"

Figure 9.13: Review extracted from the 20% subset of the reviews for which $\widehat{p} = 0.06$.

It is granted a really low $\widehat{p}$ and in this case, bringing the threshold value t this far down would mean that we predict all of the reviews in the 20% subset as discussing driver 9. 'ease of use'. It is also not naturally clear why this driver would discuss 'ease of use'. Because it is says that shaver is lightweight?

In the end, our conclusion for what we see here is that part of the annotation of reviews will always remain subjective. Anyone applying the models proposed in this thesis should ask him- or herself critically: when do we say that a review discusses driver A and when do we say it discusses driver B? Critical annotation of the reviews that are used to train a model is a large factor of influence for its performance.

9.3. Variable importance

When using a random forest method, we do not these have variables selection that the Elastic net models with $\alpha > 0$ offer. We do have the option of using variable importance though. We build a random forest model using the same 80% train set used to build the Elastic net model earlier in this chapter. We then compute the variable importance for all variables for this model. Ordering them and taking the variables deemed most important gives us the following result.

variable	importance
skin	0.0072
face	0.0071
irrit	0.0052
hair	0.0022
leav	0.0017
rash	0.0014
burn	0.0013
without	0.0013
neck	0.0012
panason	0.0011
go	0.0011
well	0.0011

(a) Random forest model

variable	estimated coefficient
neck	0.0652
irrit	3.31
skin	1.10
star	0.120
face	0.838
pai	0.890
recharg	0.747
left	0.907
rash	3.08
burn	1.85
forev	0.380
self	0.00211

(b) Elastic net model

Figure 9.14: Most important variables for the random forest model and estimated non-zero coefficients for the Elastic net model for driver 12. 'shaving irritation'.

So we see that the variables deemed important more or less correspond to the variables for which the coefficients were estimated as non-zero in the Elastic net model. We see that for both types of models we can gain insight into what words play an important role in predicting the driver outcomes.

Furthermore, note that the results shown in this chapter are all obtained in the binary setting. Results can also be obtained for models in the multinomial setting. Instead of setting a sensitivity and specificity level, we then work with the concepts of *precision* and *recall*. Setting levels for these is also equivalent to setting a misclassification matrix A. Furthermore, the Elastic net models produce a set of coefficients for each class in the multinomial setting. The random forest method will on the other hand not make this distinction, and compute variable importance for all variables as it normally does.

10

Conclusions and recommendations

The main goal of this thesis is to investigate whether we can predict driver presence and sentiment using annotated reviews. This was done both in the setting of a binary classification problem as well as a multinomial classification problem. We showed how reviews can be rewritten as a set of variables, representing words, and how to apply Elastic net and random forest models. Furthermore, research was done in how to evaluate the predictions of these models using AUC values, and how to generalize this measure to the multinomial setting.

10.1. Conclusion

The reviews in the data sets used within this thesis had been (manually) annotated for sets of drivers, where drivers represent topics. We converted the resulting driver scores to 1s and 0s to create a binary classification problem for every driver. Expanding on that, we created a multinomial classification problem by binning the driver scores into three sets, namely positive, negative and NA, where NA represents absence of the given driver.

The vocabulary in the data sets was used to create variables, where each variable represented a unique word within the data set. We thus created large numbers of variables. Reviews were then expressed as combinations of outcomes of these variables. Various function representations for these variables were considered and tested within this thesis. These include using indicator functions as a variable, giving a 1 when a review uses the corresponding word and a 0 when it does not, and also more complex metrics such as tf-idf values. We found that generally the indicator form and sometimes the normalized tf-idf form generated the best results.

We applied Elastic net and random forest models to the classification problem posed by each driver. The models were trained on generally imbalanced data sets. We showed how both types of models could be adapted to overcome the potential problems this brought. Furthermore, both models were expanded for a multinomial classification problem. From empirical results, we concluded that the random forest, specifically with our stratified sampling adaption, gave the best performance among the models evaluated.

The imbalance of our data set also posed a challenge as to how to evaluate our models. We would generally have class-imbalanced data set,s with the far larger number of reviews not discussing a driver, let alone with a specific type of sentiment. We showed how the AUC can be used as a measure to evaluate the results for the models in the binary classification problem. We then also introduced the AUC_μ as a generalisation of the AUC in the multinomial setting. These measures were used to evaluate the performance of our models in class-imbalanced settings, and results were verified in a k-fold cross validation process.

An approach was formulated to build an ANOVA model around the results, so that the results could be compared using (interaction) effects plot. Through the use of these plots, we uncovered which models performed best for the problem of this thesis. It was shown that the random forest model, specifically with our stratified sampling adaption, performed best, in combination with either and indicator or normalized tf-idf matrix form performed best.

In the end, how good we were able to predict driver scores is dependent on the application in multiple ways.

For predicting the driver scores, one needs to determine what levels of sensitivity and/or specificity are acceptable when applying the model. Then, the performance of the model is very dependent on the driver and sentiment one is looking for, and how many reviews discussing these drivers were available. For certain drivers performance can be rather poor, but generally we found our models to perform rather well, with a few extremely good outliers for certain drivers. Of course, 100% accuracy remains near impossible and is not achieved with our models, but nevertheless, we are satisfied with our results. We have managed to reach the goal with which this work was started: predicting driver presence and sentiment for reviews. We were able to quantify the performance of our models, and gained insight into how these models worked and what influenced their performance. This was done on both a large scale, looking at the number of reviews annotated for different driver and sentiment combinations, and a micro scale, looking at individual words in reviews and observing how their use influenced the model performance.

10.2. Discussion

In this section, we will discuss the various choices made in this thesis and the potential routes for further research regarding them. We will discuss the processing of the documents to variables, the use of evaluation measures, the models applied, how we compared our results and lastly, the data sets used.

10.2.1. Representing reviews as variables

In this thesis, we applied a bag-of-words model and furthermore removed stop words and stemmed words to the base, before converting the free text to variables for our model. In this process, the right choice of stop words and for the stemming process can be critical for any potential results. For example, in this thesis the choice was made to include 'not' as a stop word, but this choice is controversial. There is a difference in the meaning of 'good' and 'not good', but by removing 'not' from all reviews they become the same thing. On the other hand, many people may use the word 'bad' instead of 'not good', and the use of the word 'not' may ultimately only contribute as noise for a model. For the stemming process, choosing the right stemming algorithm can also be quite a challenge. For example, think about 'do', 'doing' and 'doings'. For our problem, we probably want these three to all mean the same thing. So remove '-ing' and '-ings' for the second and third case respectively. Now think of 'air', 'airing' and 'airings'. When discussing tv-shows, 'airing' could be interpreted as the moment of going live, something else than the 'air' that we breath. Thus, we may not want the same stemming here, as in the first example. Both the choice of stop words and stemming algorithm can be made a study on its own, and improvement here is therefore most certainly possible.

We used various forms for the variables. A direction for further research would be to expand this set of possible matrix forms. Many variations of the tf-idf matrix form that we applied can be thought of. Alternatives can be found for both the term frequency and inverse document frequency parts of tf-idf. Results can then be generated in the same manner as was done in Chapter 8 of this thesis, and compared with our matrix forms.

Furthermore, when it comes to the variables, the whole bag-of-words model can be considered a point of discussion. In language, the message of a whole review is often not fully expressed in one or multiple words. While they may give a good indication of this message, the order of the words and the punctuation used is often also important for this. Our model does not account for this. It could be extended to include punctuation (the number of exclamation marks used may for example be a strong indication of positive sentiment). Furthermore, one can think of ways of forming the variables by means other than one-to-one creating a variable for every word. A different technique known in the world of text analysis is called Word2vec[4], and forms a way of expressing a word as a numerical vector in a finite dimensional space. In this space, words can then be compared. This logic could be extended to mapping reviews to a finite dimensional space, and then we find a variable representation for our reviews again.

10.2.2. Evaluation measures

In this thesis, we used the AUC as a way to measure the performance of our models in the binary setting. In Chapter 4, we generalised the AUC to the AUC_μ for the multinomial setting. The AUC_μ as introduced in this thesis is a relatively recent adaption of an older measure that was named the K. Both measures attempt to generalise the idea that the AUC, in the binary setting, is the probability of scoring a 1 greater than a 0.

Another way to compute an AUC-like measure in the multinomial setting would be to compute a *volume under the surface* (VUS), instead of an area under the curve. This idea is not new, and has been written about before[9]. However, the proposers of the AUC_μ note that it is computationally cheaper and retains more properties of the (binary) AUC than the common VUS approaches. This remains subjective though, and the use of a different measure than the AUC_μ for the multinomial setting could be explored as a part of future research.

10.2.3. Models applied

Elastic net and random forest models have been applied in this thesis, and both have been adapted to compensate for the class imbalance we generally had in our classification problems. A point of criticism is how we determined the number of trees to grow for our random forests though. As explained in Chapter 6, we used the number of trees for which the out-of-bag error stabilises as a measure to be sure that we were growing enough trees, and often we grew far more. While this is also customarily the way to determine the number of trees to grow, we noted that use of the majority vote is not the right course of action when using the random forests for an imbalanced classification problem. We also compensated for this by lowering the threshold, and thus not using majority vote, when using the default option random forest method. We did not do this for the stratified random forest options. The error rates were always stable for the number of trees grown, but the plots revealed that the minority class would sometimes have an extremely high error rate. This does not necessarily concern us, as we are ultimately interested in the AUC and AUC_μ values given by the models, but we want to be sure that we are actually growing enough trees, and our method to see if we do seems flawed. One way of doing this, could be to use the out-of-bag votes to compute an out-of-bag AUC, or out-of-bag AUC_{mu} in the multinomial setting, and see at which number of trees that stabilizes. This would be a more elegant way to determine whether the number of trees grown is sufficient. When the AUC stabilizes, the out-of-bag error rate will also be stable. The other way around works does not work; the error rate may have stabilized, but at that point we do not know if that is also true for the AUC.

As another direction for future research, one could consider using models different than the Elastic net and random forest models. One could consider the use of a neural network or support vector machine model. Not all models may lend themselves immediately for the problem statement in this thesis, for we need a probability as an outcome, so that we can compute an AUC. One could then (attempt to) adapt these models such that they can.

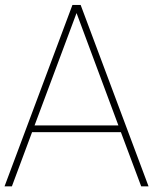
10.2.4. Data sets

We used two data sets of internet reviews in this work. In Chapter 9 it was shown that there are inconsistencies in how reviews are annotated. One way to improve the results would be to remove any such inconsistencies from the data set beforehand. How a review should be annotated is subjective though, making one hundred percent consistency an impossibility. Furthermore, it was shown in Chapter 8 that the performance of models increased significantly as we increased the size of the data set. Lastly, our analysis was now carried out using data on shavers. It could be that when this analysis is performed for data sets in a different domain, the results change as well, due to a change in the vocabulary and characteristics of the review.

10.3. Further recommendations

The goal of this thesis, predicting driver scores for reviews, has to some extent been achieved. 100% accuracy is an impossibility, but with our models we can give a rather good guess. One question that remains is whether the models can be used to automate the annotation tool the annotation process itself. For this application, one would have to further specify the models, by setting threshold values t or choosing misclassification matrices A. The predictions generated could then help the annotation process by functioning as a proposition for how the review should be annotated. This could already significantly speed up annotation of the reviews, as it becomes a matter of accepting and rejecting predicted driver scores. Additionally, words deemed relevant for the prediction of drivers could be highlighted in the review text. This would give the person annotating reviews insight into why a proposition for a driver score is made, and show the person where in the review to look for the relevant piece of text.

Note though that the driver scores in the data (usually) range from -2 to 2 (with a negative score meaning negative sentiment, and a positive score meaning positive sentiment), or are NA. Our models have only gone as far as being able to predict positive sentiment, negative sentiment or NA. Thus an implementation of the model in the current annotation tool could only function as a means to give an indication to someone annotating reviews. This, or the annotation tool could be adapted to only distinguish positive sentiment, negative sentiment and NA for each driver. Then, our model could give the predictions fully automated, although it would make mistakes of course. Further research into better models using ideas from the discussion could be done to tackle this. In the end, how to continue from here on depends on the wishes of a potential application.



List of stop words

i	these	he'd	where's	on
me	those	she'd	why's	off
my	am	we'd	how's	over
myself	is	they'd	a	under
we	are	i'll	an	again
our	was	you'll	the	further
ours	were	he'll	and	then
ourselves	be	she'll	but	once
you	been	we'll	if	here
your	being	they'll	or	there
yours	have	isn't	because	when
yourself	has	aren't	as	where
yourselves	had	wasn't	until	why
he	having	weren't	while	how
him	do	hasn't	of	all
his	does	haven't	at	any
himself	did	hadn't	by	both
she	doing	doesn't	for	each
her	would	don't	with	few
hers	should	didn't	about	more
herself	could	won't	against	most
it	ought	wouldn't	between	other
its	i'm	shan't	into	some
itself	you're	shouldn't	through	such
they	he's	can't	during	no
them	she's	cannot	before	nor
their	it's	couldn't	after	not
theirs	we're	mustn't	above	only
themselves	they're	let's	below	own
what	i've	that's	to	same
which	you've	who's	from	so
who	we've	what's	up	than
whom	they've	here's	down	too
this	i'd	there's	in	very
that	you'd	when's	out	will

B

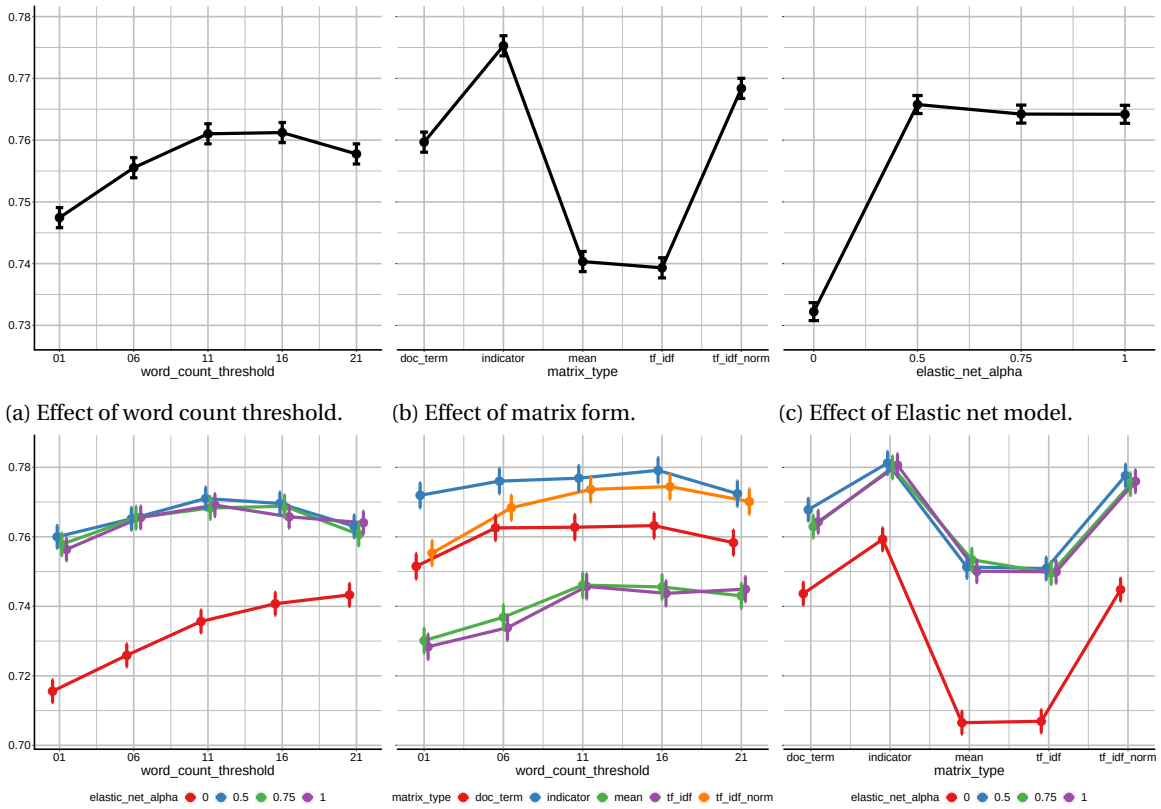
Results using a different misclassification matrix

Matrix A is chosen as

prediction(↓), true value (→)	P	N	NA
P	0	3/5	6/5
N	3/5	0	6/5
NA	6/5	6/5	0

Figure B.1: Choice of matrix A

Elastic net models



(a) Effect of word count threshold.

(b) Effect of matrix form.

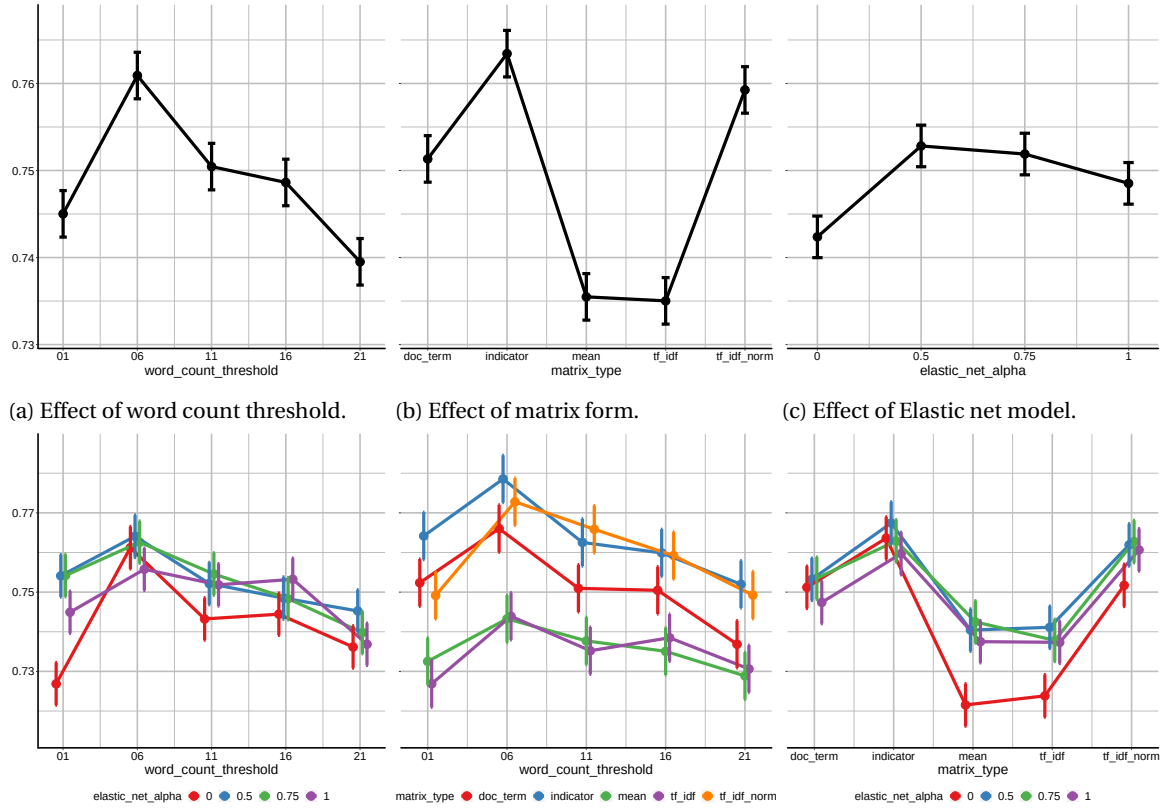
(c) Effect of Elastic net model.

(d) Effect of word count threshold vs. Elastic net model.

(e) Effect of word count threshold vs. matrix form.

(f) Effect of matrix form vs. Elastic net model.

Figure B.2: Main effects plots and interaction effects for the effects of the matrix form, word count threshold and Elastic net models. On the y-axis is the AUC_μ value, on the x-axis the (first) varying parameter for its respective effects plot. Results were obtained using the $n1976$-data set. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.



(a) Effect of word count threshold.

(b) Effect of matrix form.

(c) Effect of Elastic net model.

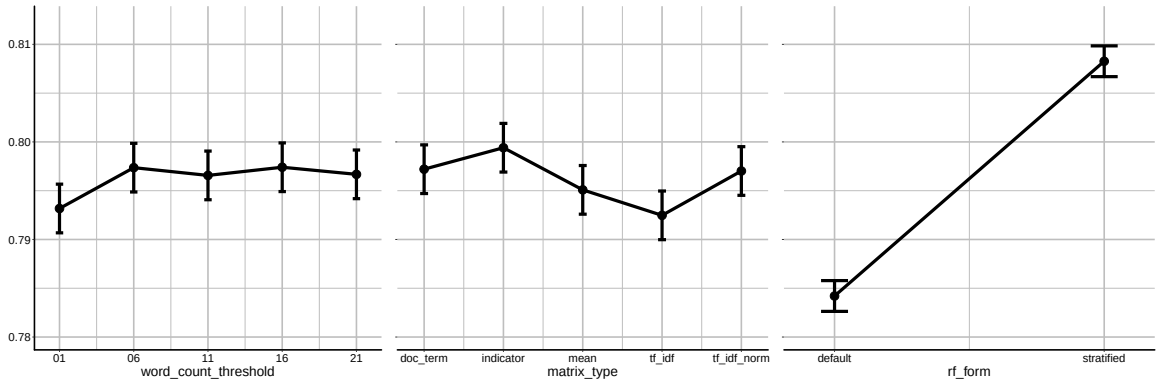
(d) Effect of word count threshold vs. Elastic net model.

(e) Effect of word count threshold vs. matrix form.

(f) Effect of matrix form vs. Elastic net model.

Figure B.3: Main effects plots and interaction effects for the effects of the matrix form, word count threshold and Elastic net models. On the y-axis is the AUC_μ value, on the x-axis the (first) varying parameter for its respective effects plot. Results were obtained using the $n775$-data set. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

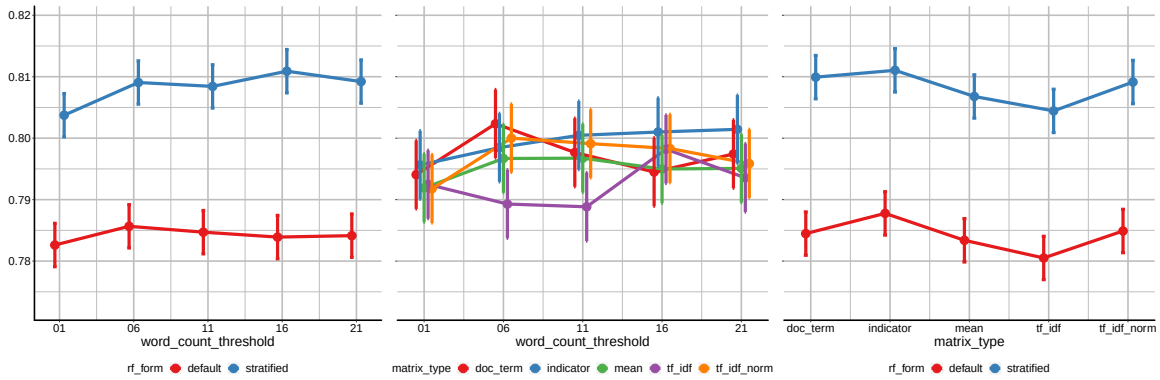
Random forest models



(a) Effect of word count threshold.

(b) Effect of matrix form.

(c) Effect of Elastic net model.

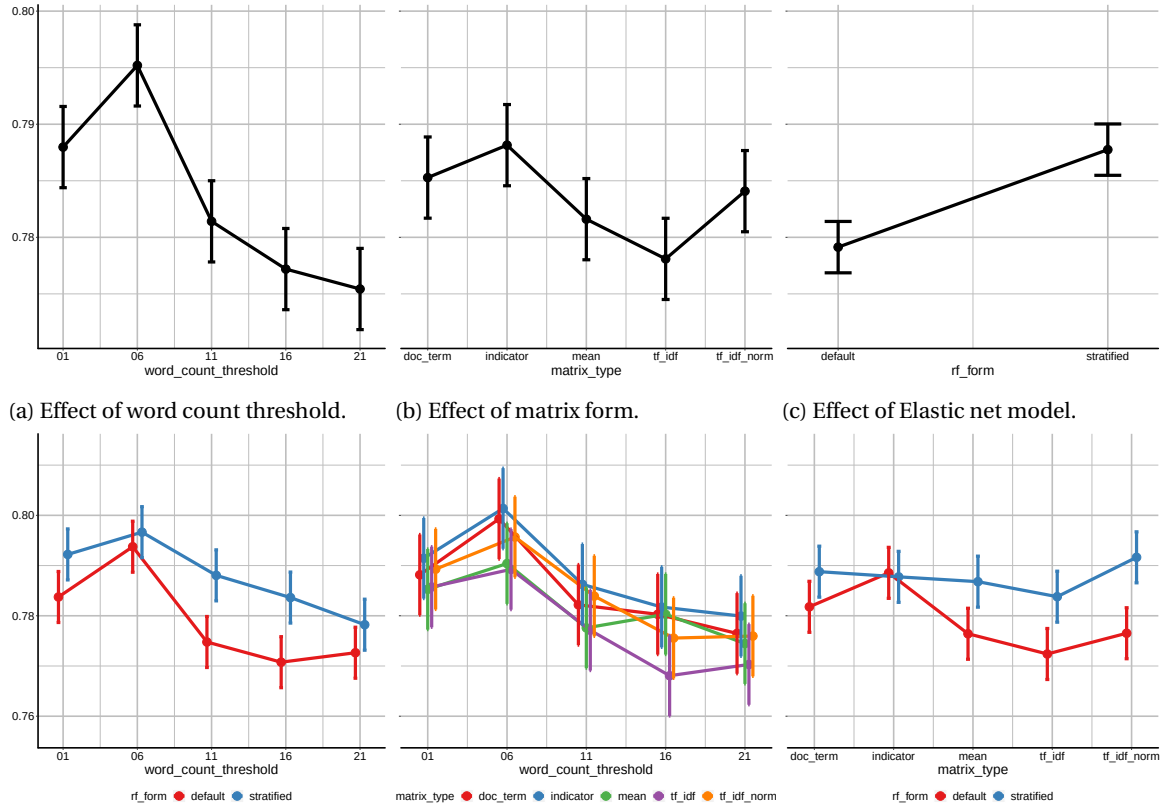


(d) Effect of word count threshold vs. Elastic net model.

(e) Effect of word count threshold vs. matrix form.

(f) Effect of matrix form vs. Elastic net model.

Figure B.4: Main effects plots and interaction effects for the effects of the matrix form, word count threshold and Elastic net models. On the y-axis is the AUC_μ value, on the x-axis the (first) varying parameter for its respective effects plot. Results were obtained using the $n1976$-data set. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.



(a) Effect of word count threshold.

(b) Effect of matrix form.

(c) Effect of Elastic net model.

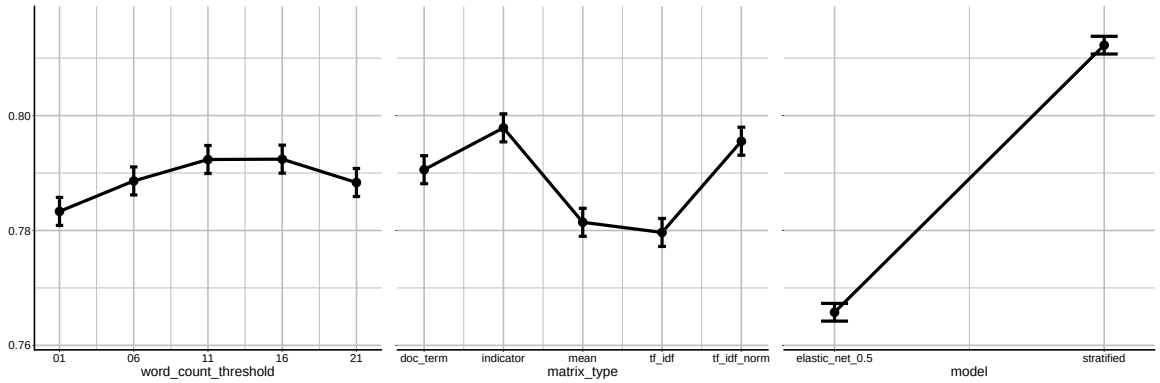
(d) Effect of word count threshold vs. Elastic net model.

(e) Effect of word count threshold vs. matrix form.

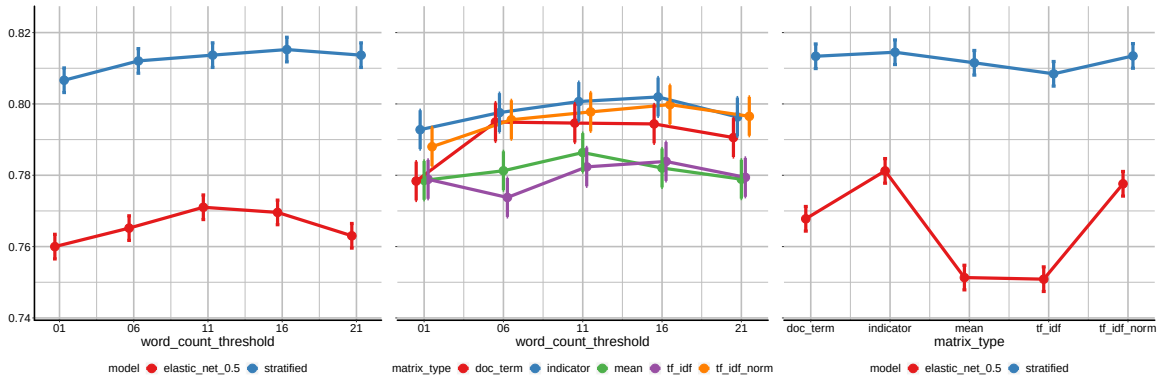
(f) Effect of matrix form vs. Elastic net model.

Figure B.5: Main effects plots and interaction effects for the effects of the matrix form, word count threshold and Elastic net models. On the y-axis is the AUC_{μ} value, on the x-axis the (first) varying parameter for its respective effects plot. Results were obtained using the $n775$-data set. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

Comparison Random forest and Elastic net

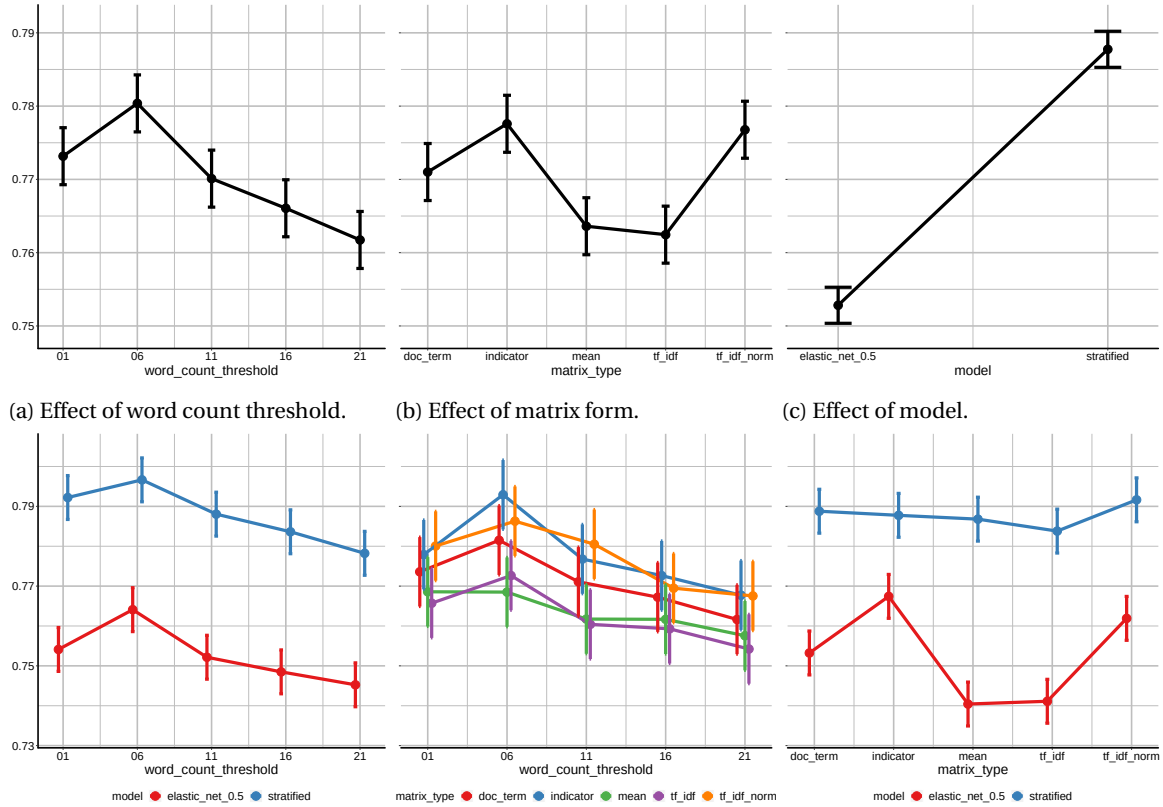


(a) Effect of word count threshold. (b) Effect of matrix form. (c) Effect of model.



(d) Effect of word count threshold vs. model. (e) Effect of word count threshold vs. matrix form. (f) Effect of matrix form vs. model.

Figure B.6: Main effects plots and interaction effects for the effects of the matrix form, word count threshold and Elastic net models. On the y-axis is the AUC_μ value, on the x-axis the (first) varying parameter for its respective effects plot. Results were obtained using the $n1976$-data set. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.



(a) Effect of word count threshold.

(b) Effect of matrix form.

(c) Effect of model.

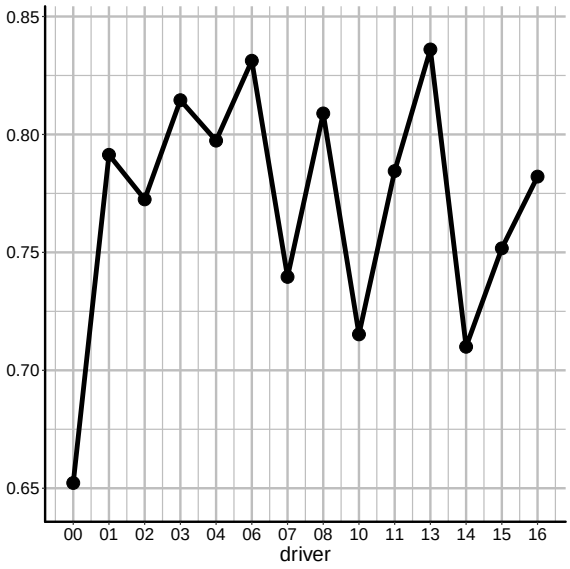
(d) Effect of word count threshold vs. model.

(e) Effect of word count threshold vs. matrix form.

(f) Effect of matrix form vs. model.

Figure B.7: Main effects plots and interaction effects for the effects of the matrix form, word count threshold and Elastic net models. On the y-axis is the AUC_{μ} value, on the x-axis the (first) varying parameter for its respective effects plot. Results were obtained using the $n775$-data set. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

Driver results



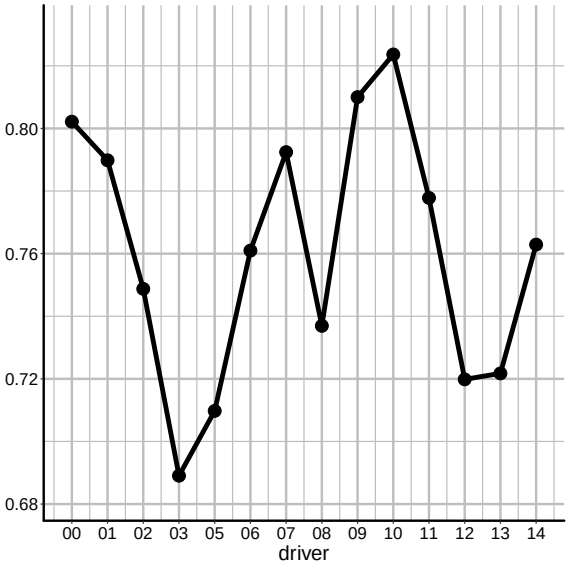
(a) Effect of driver number.

0.	Shaving (general)
1.	During shaving – speed
2.	During shaving – comfort
3.	After shaving – irritation
4.	After shaving – closeness
5.	Cofo
6.	Sound
7.	Ease of use
8.	Cleaning
9.	Smart Clean
10.	Service & Delivery
11.	Accessories
12.	Features
13.	Charging & electronic power
14.	Design
15.	Value for money
16.	Issues & reliability

(b) Drivers corresponding to the driver numbers in the effect plots.

Figure B.8: Effects plot for the effect of driver number. On the y-axis is the AUC value, on the x-axis the varying parameter for the driver number. A list of drivers corresponding to the driver numbers is provided. Results were obtained using the n1976-data set running elastic net models for all levels of α and both random forest options at all word count thresholds for all matrix types. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

For the n775-data set



(a) Effect of driver number.

1.	Accessories
2.	Charging & Electronic power
3.	Design
4.	Features
5.	Issues & Reliability
6.	Service & Delivery
7.	Value for money
8.	cleaning
9.	ease of use
10.	shaving closeness
11.	shaving comfort
12.	shaving irritation
13.	shaving rest
14.	shaving speed
15.	smart clean

(b) Drivers corresponding to the driver numbers in the effect plots.

Figure B.9: Effects plot for the effect of driver number. On the y-axis is the AUC value, on the x-axis the varying parameter for the driver number. A list of drivers corresponding to the driver numbers is provided. Results were obtained using the n775-data set running elastic net models for all levels of α and both random forest options at all word count thresholds for all matrix types. Results were obtained using a 5-fold cross validation. $l = 1$ computation was run.

Bibliography

[1] L. Breiman. Random forests. *Machine Learning*, pages 45, pp. 5–32, 2001.

[2] Leo Breiman Chao Chen, Andy Liaw. Using random forest to learn imbalanced data. *UC Berkely*, 2004.

[3] Janitza et al. An auc-based permutation variable importance measure for random forests. *BMC Bioinformatics*, pages 14, article number:119, 2013.

[4] Mikolov et al. Computing numeric representations of words in a high-dimensional space. URL `https://patents.google.com/patent/US9037464B1/en`.

[5] Yulan He Harith Alani Hassan Saif, Miriam Fernandez. On stopwords, filtering and data sparsity for sentiment analysis of twitter. *The 9th International Conference on Language Resources and Evaluation*, 2014.

[6] Tin Kam Ho. Random decision forests. *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, pages pp. 278–282, 1995.

[7] Robert Tibshirani Jerome Friedman, Trevor Hastie. Regularization paths for generalized linear models via coordinate descent. *J Stat Softw*, pages 33(1), pp. 1–22, 2010.

[8] Ketan Shah Kranti Vithal Ghag. Comparative analysis of effect of stopwords removal on sentiment classification. *International Conference on Computer, Communication and Control (IC4)*, 2015.

[9] Jason P. Li, Jialiang; Fine. *ROC analysis with multiple classes and multiple tests: methodology and its application in microarray studies*. Oxford University Press, 2008.

[10] Douglas C. Montgomery. *Design and analysis of experiments 8th edition*. John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ, 2013.

[11] Gupta M. R. O'Brien, D. B. and R. M. Gray. Cost sensitive multi-class classification from probability estimates. *Proceedings of the 25th international conference on Machine Learning-ICML'08*, pages 712–719, 2008.

[12] M.F. Porter. Snowball: A language for stemming algorithms. URL `https://pdfs.semanticscholar.org/0d8f/907bb0180912d1e1df279739e45dff6853ee.pdf?_ga=2.84929820.404445213.1589298510-981098873.1588798030`.

[13] J.D. Rajaraman, A.; Ullman. Data mining. *Mining of Massive Datasets*, pages 1–17, 2011.

[14] David Page Ross S. Kleiman. AUC_μ: A performance metric for multi-class machine learning models. *Proceedings of the* 36^{th} *International Conference on Machine Learning*, 2019.

[15] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, pages Vol. 58, 267–88, 1996.

[16] David J. Hand Robert J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning, 45*, pages 171–186, 2001.

[17] Jerome Friedman Trevor Hastie, Robert Tibshirani. *The Elements of Statistical Learning 2nd edition*. Springer Series in Statistics, Stanford, California, 2017.

[18] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, pages Vol. 67, Part 2, pp. 301–320, 2005.