Complaint Handling at the Dutch National Police

Using machine learning to (partly) automate the handling of online trade fraud complaints

Author:
William Kos [3933083]
E-mail: w.h.kos@uu.nl
Utrecht University

Supervisors
Dr. F.J. Bex
E-mail: f.j.bex@uu.nl
Utrecht University

Dr. M.J.S. Brinkhuis
E-mail: m.j.s.brinkhuis@uu.nl
Utrecht University

Dr. M.P. Schraagen
E-mail: m.p.schraagen@uu.nl
Utrecht University

Master Thesis Business Informatics

Universiteit Utrecht

POLITIE

Abstract

The Dutch National Police maintains an online interface that allows civilians to report their complaints regarding trade fraud over an online medium (e.g. eBay). Since an increasing amount of complaints are being filed, it is desirable to make an automatic distinction between complaints worth investigating and those not worth investigating. One valuable distinction which can be made early in the process is that between a complaint which will be withdrawn by either the complainant or the police and a complaint that will not be withdrawn. This thesis examines whether either one of nine machine learning classifiers trained on free text complaint data can be used for this purpose. Complicating this task is the class distribution in the data, where a majority of 86.7% is labelled as "not withdrawn". To prevent this skewness from affecting classifier performance, resampling, word weighting, and word normalization are applied, of which the influence on the classification performance is assessed.

This research shows that using machine learning, it is possible to create such a distinction by classifying complaints on whether they will be withdrawn or not. Overall, it is found that probabilistic classifiers (i.e. naive Bayes) have the highest unimproved performance and that through data alterations the performance of an optimized machine learning technique (i.e. SVM) can be improved up to 13.5 percentage points. Furthermore, by optimizing the classifiers, the difference in performance between the best classifier (i.e. Logistic regression) and the worst classifier (i.e. K-nearest-neighbor) can be reduced from 11.8 to only 4.2 percentage points.

Acknowledgements

Eight months I had the possibility to perfect the explanation of my thesis when anyone interested asked me about it. What started out as a fuzzy story causing people to have more questions at the end of it than before, has turned into something like a fairytale. And now, after eight months, this fairytale gets its own perfect ending: “And William lived happily ever after”. Until I start working of course and realize that studying wasn’t so bad after all, but hey, that’s in the future right?

I would like to thank my first supervisor, Floris Bex, for helping me out during my thesis and guiding me when I was lost, even though we did not always agree and I was sometimes a bit stubborn. Secondly, I would like to thank Matthieu Brinkhuis for his feedback during the data analysis and final stage of my thesis. Thirdly, I want to thank Marijn Schraagen for sharing his knowledge on machine learning, based on which this research was conducted. Additionally, I want to thank Ben van Scheppingen and Bas Testerink, who enabled me to get in touch with the right contacts at the Dutch Police.

At last, I want to thank my parents and friends, who have supported me throughout my entire thesis and kept me motivated when I needed it.

William Kos

Utrecht, June 2017

Table of Contents

Table of Figures

Table of Tables

1 Introduction

The internet has given criminals new ways to conduct their activities. For example, many trade fraud cases now occur online (e.g. fake designer goods which are sold as real designer goods on the internet). Online trade fraud can be defined as the committing of a scam over the internet (Leukfeldt, Domenie & Stol, 2010). A rising trend in online trade fraud can be noticed in the Netherlands where in 2013 3.3% of the Dutch citizens were the victim of online trade fraud as opposed to 2.9% in 2012 (Inspectie Veiligheid en Justitie, 2015). To stop the rising trend, the Dutch police is forced to come up with new solutions for tackling online trade fraud.

The police has devised a solution to tackle online trade fraud. Since criminals who conduct their business online are not limited to geographical areas, it is desirable and essential to have a central point where the trade fraud complaints are aggregated and dealt with. For this the Dutch police, in 2010, established a unit responsible for solving online trade fraud cases, LMIO[1]. To further improve the intake and registration of online crime, and the collaboration of the police and the commercial parties, the police formed the National Service Center e-Crime, LSCeC[2], as an extension of LMIO in May 2015 (Streefkerk, 2015).

To ensure LMIO thrives as a project, its success and workflow have recently been evaluated (Inspectie Veiligheid en Justitie, 2015). This evaluation showed that LMIO succeeds at being the central point it was intended to be, handling approximately 90% of all online trade fraud complaints. Furthermore, this evaluation showed that LMIO's way of working follows the general process the Dutch police follows in their cases (de Poot et al., 2004). Complaints received through www.politie.nl are analyzed by crime analysts according to intuition and police guidelines to determine which are relevant for creating a dossier. Such a dossier consists of a story and evidence and is built up by police planners through investigatory actions in which obtained information is used to either prove or disprove hypotheses (van den Braak, 2010; Latukolan & van Ginkel, 2016; de Poot et al., 2004). Dossiers created by LMIO are used as input by local police units to start a case.

Approximately 50.000 complaints are filed online each year, with the average loss per complaint estimated at 200 euros (Streefkerk, 2015). Employees at DLOC[3], an administrative police unit that handles incoming data, register these complaints in the enforcement database, BVH[4], and analyze them for completeness and usability. The outcome of this analysis is communicated through e-mail to LMIO, where it is used to refine their own analysis (Peters, 2016b). However, since LMIO does not have enough manpower, the online trade fraud complaints cannot all lead to a case (Inspectie Veiligheid en Justitie, 2015). It is thus important

[1] Landelijk Meldpunt Internet Oplichting
[2] Landelijk Service Centre e-Crime
[3] Dienst Landelijk OperatieCentrum
[4] Basisvoorziening Handhaving

that a distinction is made as early as possible between complaints worth investigating and those not worth investigating, so that resources are not wasted. To ensure missing information does not play a role in this distinction, Bex et al. are currently creating an agent, capable of understanding free-text fields using NLP techniques, to communicate with online trade fraud victims to obtain all information necessary for building a case while automatically processing it (Bex, Peters, & Testerink, 2016).

To assist crime analysts in the distinction between complaints worth investigating and those not worth investigating machine learning techniques could be used. Machine learning techniques are increasingly being applied for classification purposes in the field of crime analysis (Chen et al., 2004; Sharma & Panigrahi, 2012). Research in this field explores and compares the use of many classifiers including naive Bayes, decision trees, and neural networks (Abdelhamid et al., 2014; Baumgartner et al., 2008; Gupta et al., 2016; Oatley & Ewart, 2003). Since the dataset used in this research consists of both textual high degree of skewness, research on classification for textual data will have to be combined with research on classification for skewed data to find the proper distinctions on complaints worth investigating and those not worth to assist both LMIO's crime analysts and employees at DLOC (Brause et al., 1999; Japkowicz & Stephen, 2002; Phua et al., 2004; Sebastiani, 2002; Sun et al., 2007).

An example of a valuable distinction which can be made early in the process is that between a complaint for a civil case and one for a criminal case. In a civil case, a buyer has received an item which did either not live up to the expectations or broke down after some days. The delivery of such items is not considered a criminal act, article 326 of the Criminal Code[5] does not apply. Even though the online form for filing complaints uses a decision tree to distinguish civil from criminal cases and refers civil complaints to a foundation which provides free juridical advice[6], LMIO still receives civil complaints. It often occurs that such complaints are first taken up for investigation and discarded later in the investigation, after resources have been spent on them.

The purpose of this research is to investigate how machine learning techniques can be utilized to automatically distinguish online trade fraud complaints worth investigating from those not worth investigating and thereby helping both LMIO's crime analysts and employees at DLOC in their daily job. Peters (2016a) has already performed a preliminary study to determine which machine learning techniques could be used to distinguish online trade fraud complaints, and after applying these techniques Peters has obtained initial promising classification results. However, by increasing and improving the machine learning techniques used, better results regarding complaint distinction can hopefully be obtained, so that a

[5] Wetboek van Strafrecht
[6] Juridisch Loket

resulting classifier can be used by the Dutch police to (partly) automate and thereby support the handling of online trade fraud complaints. The main research question is as follows:

"How can the handling of online trade fraud complaints be (partly) automated?"

In order to answer the main research question, a set of subsequent research questions has to be answered:

- What are the characteristics of complaint data? (Chapter 4 & 6)
- Which steps in the process of handling a complaint can be (partly) automated given the current data availability? (Chapter 3, 4, & 5)
- Can machine learning techniques be utilized for complaint handling? (Chapter 6)
- What machine learning techniques perform best with respect to the automatable steps in the process of handling a complaint? (Chapter 7 & 8)

This master thesis consists of nine chapters. The next chapter contains the method that has been followed in this research. In chapter 3 it is described what criminal investigation in general is and how it has been applied in the Netherlands. Chapter 4 explores the handling of an online trade fraud complaint by both DLOC and LMIO, after which chapter 5 explores how the process of handling an online trade fraud complaint can benefit from data mining. Chapter 6 describes the theory behind data mining and how it can be applied in this research. In chapter 7, the conducted experiment and obtained results are set out, after which the results are discussed in chapter 8. The research questions set out above are answered in chapter 9 and finally, a set of ideas and improvements for future research are presented in chapter 10.

2 Research Approach

In this research an initial and extensive data analysis has been performed to analyze the performance of machine learning techniques on classifying online trade fraud complaints using a design science approach (Wieringa, 2010). This research has focused on the creation of a binary classifier for textual skewed data. To discuss and confirm the findings of the data analyses, interviews have been held with both LMIO's crime analysts and employees at DLOC. This research consisted of the following steps, which are also depicted in Figure 1:

1. First, an extensive literature study has been conducted to obtain knowledge on the way of working at the police. The theoretical knowledge obtained from this has served as the basis upon which the remainder of this research has been built. The goal was to understand, from a theoretical perspective, how LMIO's analysts and employees at DLOC handle complaints so that this research could be adjusted to serve their actual daily pursuits.

2. After knowledge had been obtained on the way of working, a literature study has been performed to acquire knowledge on the use of machine learning techniques. In this study the observed techniques used in crime analysis has been compared with respect

to their applicability for complaint classification. Next to this, a separate literature study has been performed to determine which machine learning techniques are most appropriate for classifying either textual or skewed data. The result of these studies was a list of machine learning techniques, along with their characteristics. The performance of the machine learning techniques on this list have been analyzed in the data analysis.

3. To ensure this research met the needs of both LMIO's crime analysts and employees at DLOC, they were interviewed on their view of complaint classification. For this, the complaint data was first examined to obtain in-depth knowledge on its characteristics. Based on these interviews, an ordered list of useful classification metrics has been created, which was used to initiate the data analysis.
4. As mentioned in Chapter 1, previous research on complaint classification has already been performed by Peters (2016a). For this the Waikato Environment for Knowledge Analysis, WEKA (Frank et al., 2010), was used, which is a machine learning tool kit. Performed actions and found results were analyzed to assist as a starting point for the remainder of the data analysis.
5. After all preparatory work was completed, the actual data analysis began. A process was gone through in which each machine learning technique found in step 2 was applied on the dataset to determine whether classification on textual skewed data could be appropriately achieved for the top classification metric resulting from step 3. For each machine learning technique, the influence of data alterations on its classification performance was assessed, after which the classification results were compared to a baseline set through both experience and discussions with LMIO's crime analysts and employees at DLOC.

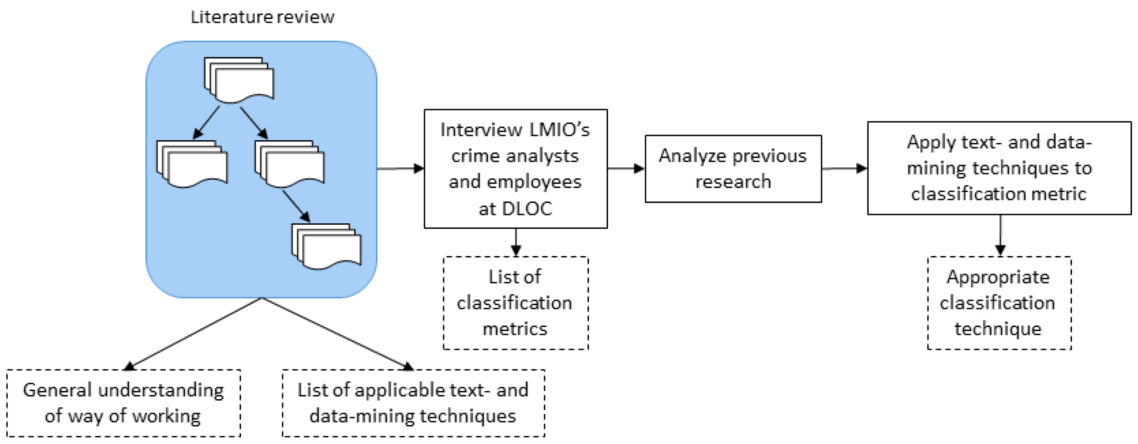

Literature review
Interview LMIO's crime analysts and employees at DLOC
Analyze previous research
Apply text- and data-mining techniques to classification metric
List of classification metrics
Appropriate classification technique
General understanding of way of working
List of applicable text- and data-mining techniques

Figure 1: Research approach

3 Analyzing Criminal Investigation

This chapter aims to provide a general understanding of what criminal investigation is and how it has been practiced in the Netherlands. It discusses the theory behind crime analysis and the different steps a Dutch crime analyst undertakes to solve a case.

3.1 The theory behind criminal investigation

In the literature, criminal investigation is generally seen as the overarching term for a discipline which can be broken down into a multitude of different sub-disciplines (Boba, 2005; Osborn and Wernicke; 2003). For example, in the Netherlands a distinction is made between on the one hand operational or tactical (de Poot et al., 2004) and on the other hand strategic crime investigation (van den Braak, 2010). Operational crime investigation is focused on supporting the investigation and solving of a single case, while strategic crime investigation tries to prevent crime through trend prediction. An example of operational crime investigation could be identifying and arresting a burglar, while with strategic crime investigation the pattern of burglaries in a neighborhood is used to predict when and where the next burglary will occur. De Poot et al. (2004) also make the distinction between reactive and proactive investigation; the former starts with an identified crime, for instance after a complaint, while the latter starts with a crime suspicion about certain persons or companies based on available information.

Standing at the center of criminal investigation is the act of problem solving. Problem solving involves an initial state, a final state, and a set of actions to overcome the distance between the states. These actions are based upon previous experiences which are stored in schemas, a knowledge structure in which thoughts, behavior, and the relationships among them are categorically arranged (DiMaggio, 1997). For police investigators or crime analysts these experiences are obtained in previous cases. Pirolli and Card describe schemas for police investigators or crime analysts as “a set of patterns around the important elements of their tasks” (Pirolli and Card, 2005, p. 1). In criminal investigation schemas can be triggered by case characteristics (e.g. type of murder), as well as by the case goal. When, for example, the goal of a case is to inform rather than prosecute, a different schema is activated (de Poot et al., 2004). The ultimate goal of each case is to solve it, which depends upon the distance between the initial and the final state, the problem situation and case goal. This distance is considered small when at the start of the investigation it is already clear how this goal can be reached, and considered large when this is unclear and there are many sub goals and actions which have to be undertaken to reach the case goal (de Poot et al., 2004).

In order to reach the case goal and solve a case, a story has to be (re)constructed. De Poot et al. (2004) mention a story consists of “the circumstances which can be seen as causes and reasons for acts resulting in consequences”[7] (p. 44). Criminal investigations consist of a reconstruction phase, in which (part of) the story is reconstructed, and a verification phase, in which evidence is gathered to support the story (de Poot et al., 2004). During the investigation these phases interact in an iterative process: stories are constructed and subsequently verified or falsified using arguments and evidence (Bex, 2011). According to Crombag, van Koppen, and Wagenaar (1994) the quality of the reconstructed story accounts for half of the proof in a case. The other half of the proof is formed by the manner in which the reconstructed story is anchored using evidential data found during the verification phase.

3.2 Dutch criminal investigation

Based upon the existence of both a suspect and a story four types of cases can be discerned (de Poot et al., 2004):

- Crystal-clear cases: both a suspect and a story are present;
- Verification cases: a suspect’s identity and a story are present;
- Investigation cases: no suspect, but a story is present;
- Search cases: neither a suspect nor a story are present.

Not every case the police encounters is investigated. For investigation and search cases, Greenwood and Priscilla (1975) found that a dead end in the most obvious investigation tracks often results in ceasing a case. De Poot et al. (2004) have researched how often and why the Dutch police ceases cases. They found that crystal-clear cases (94%) are taken into investigation most often, followed by verification cases (72%), investigation cases (72%), and search cases (38%). In addition to the aforementioned distance between the problem situation and case goal, the policies of the police and/or Public Prosecution Service, the structural characteristics of a case, and the ease of solving it are important selection criteria. For verification and investigation cases respectively 5% and 8.5% of the cases are ceased due to the quality of the story. According to the theory of anchoring narratives (Crombag et al., 1994) this reduced quality increases the distance between the problem situation and case goal, thus requiring the police to spend more resources on recovering the story before they can start searching for evidence. Search cases, with the exception of severe cases, are most often (60%) ceased due to policies of the police and/or Public Prosecution Service, where the chance of solving the case does not weigh up against the costs.

For the cases that are taken into investigation the Dutch police uses two investigation approaches; concurrent and sequential investigation. Verification, investigation, and search cases are often investigated using a sequential approach, in which one investigation track at a time is given attention. With crystal-clear and severe cases, a concurrent approach is applied in which multiple investigation acts are carried out simultaneously, even though this could

[7] Translated from Dutch

result in performing acts which were not necessary to achieve the case goal. Cases which the police are alerted to within 15 minutes are often responded to quickly using a lot of manpower. By doing so, the police are capable of obtaining as much evidence as possible while it is still fresh and turn a case into a verification or crystal-clear case, improving the chances of it being solved quickly (de Poot et al., 2004).

During an investigation, the Dutch police uses checklists containing in more or less detail all acts which have to be performed. These checklists differ depending on the type of case. An example could be a checklist for a murder case including the three themes: [1] initial actions, [2] crime scene, and [3] neighborhood canvass. At the start of the investigation multiple initial actions have to be performed (e.g. obtain map of the area), before the crime scene can be investigated and a canvass can be held. For each of the themes an undefined number of actions is listed which have to be performed (e.g. interview cab drivers for the canvass). When the police receive a story at the start of the case, as is with verification and investigation cases, this story stands at the center of the investigation. A top-down approach is applied to either confirm, complete or alter the story. With search cases, when no story is available, the investigation and corresponding checklists are initially focused towards identifying a suspect, before a story is reconstructed (de Poot et al., 2004).

A case can have two possible outcomes:

1. The story has been reconstructed according to the Golden W's: Who is it about? What did happen? Where did it take place? By what means did it happen? In what way did it happen? When did it take place? Why did it happen? (Gross, 1908), and is supported by the evidence, after which the case is handed over to the Public Prosecution Service, or
2. The reconstruction of the story has stranded due to a lack of evidence to support the existing hypotheses and the case is ceased.

Independent of the outcome, three general actions should be and are often executed while finishing a case (de Poot et al., 2004): [1] report all information in the dossier and archive this dossier, [2] file additional information unrelated to the case, and [3] evaluate the case.

3.3 The role of the crime analyst

Crime analysis is "the identification of and provision of insight into the relationship between crime data and other potentially relevant data with a view to police and judicial practice"[8] (van den Braak, 2010, p. 13-14). In the Netherlands two types of crime analysts can be discerned (Timmers, 2016). An operational crime analyst is often assigned to a case to create order in the bulk of information retrieved during the investigation. Using the findings of an operational

[8] Original source Minnebo (2004). Translated from Dutch.

crime analyst, a tactical crime analyst tries to ultimately come to a most probable story. To determine this most probable story, the general course of action for Dutch crime analysts is to first create various hypotheses based on features of the crime and construct scenarios based on those hypotheses (van den Braak, 2010). The scenarios are initially based on the facts which are considered definitely proven. These facts are then plotted on a timeline, which is known as anchoring in time (de Poot et al., 2004). To fill the unknown gaps between the anchors, unsupported events are hypothesized and added to the scenarios. Through an iterative process of finding new evidence, looking over the evidence, and creating support for or against the hypothesized events, scenarios are either altered, deleted, or given more body (i.e. anchored), until a definitive story is reached. This process, as described by van den Braak (2010), aligns with the steps in the analysis process by Pirolli and Card (2005) in which evidence is gathered and looked over in a foraging loop, and tied to hypotheses in a sense making loop.

Choosing amongst hypotheses and determining their validity is undoubtedly the ultimate responsibility of a crime analyst. Van den Braak (2010) stresses the importance of inter-hypothesis comparison over single hypothesis evaluation. However, Heuer (1999) states that most crime analysts follow the single hypothesis evaluation approach based on their instinct, referred to as a satisficing strategy. Next to this, Heuer, along with Pirolli and Card, state that this evaluation process is restricted by human limitations; specifically, three problem domains may be distinguished (Heuer, 1999; Pirolli and Card, 2005): the confined human working memory, a perceptual prejudice along with the inability to define more than one hypothesis, and a confirmation prejudice. To overcome these limitations, Heuer (1999) proposes the use of an eight-step method, known as the analysis of competing hypotheses. In this method, all possible hypotheses along with evidence to prove or disprove them are grouped in an evidence matrix. Through a process of comparing both existing and newly acquired evidence against each hypothesis in the matrix, impossible hypotheses can be refuted until a single hypothesis remains. During this process, any accepted hypothesis should be reflected against the validity of the evidence. In the end, the remaining and refuted hypotheses should all be presented, along with the argumentation for conclusions drawn. This analysis of competing hypotheses, as proposed by Heuer, has been adopted by the Dutch crime analysts for reconstructing the story (Minnebo, 2004).

4 Handling Online Trade Fraud

In this chapter, the process of handling an online trade fraud case by both LMIO and DLOC will be explained. It describes the legislation used for prosecuting fraudsters and discusses the types of fraud which are committed. The information in this chapter is mostly derived from Latukolan and van Ginkel (2016) and interviews held at LMIO and DLOC.

4.1 Prosecuting online trade fraud

Online trade fraud cases are penalized just as regular trade fraud cases, according to article 326 of the Dutch Criminal Code. In this article, four scam modes are given which, when individually or jointly applied to produce a benefit for a fraudster, will lead to a conviction for trade fraud, namely: [1] the adoption of a false name, [2] the adoption of a false role, [3] tricky maneuvers, and [4] a contexture of figments (Bernklau & van der Putte, 2015). A tricky maneuver, for example, occurs when a fraudster sends a box of potatoes instead of the purchased item. With a contexture of figments, both the amount and intrusiveness of a fraudster's false statements are of importance. When a victim could have known he was being scammed, a contexture of figments is not in place. For LMIO, it is important that dossiers sent to the local police units consist of a story and corresponding evidence proving the occurrence of one or several of those scam modes.

4.2 Online trade fraud scenarios

The online trade fraud complaints that LMIO receives can be roughly discerned into four scenarios: [1] classic trade fraud, [2] triangular trade fraud, [3] screenshot payments, and [4] spoofed websites (Streefkerk, 2015). With respect to classic trade fraud, two types of cases can be distinguished, which are depicted in Figure 2 and Figure 3; the case in which an item is offered for sale by a fraudster but not delivered (Figure 2), and the case in which a fraudster responds to a wanted advertisement (Figure 3). Triangular trade fraud involves a fraudster buying a real item and meanwhile selling a fake item. The buyer of the fake item will be given the account number of the original seller, resulting in the fraudster receiving the real item for free, leaving the buyer empty handed and feeling betrayed by the original seller. With screenshot payments a fraudster intends to convince the seller that he transferred the money for an item through modifying a bank account screenshot and so receiving the item without actually paying for it. Spoofed websites are websites which feel genuine, through the use of fake logos and quality marks, but do not deliver bought items. With all of the four scenarios, Streefkerk (2015) mentions fraudsters making use of society trends to increase the likeliness of their product being seen by potential victims, for example by switching from standard commerce sites (e.g. Marktplaats.nl) to social media.

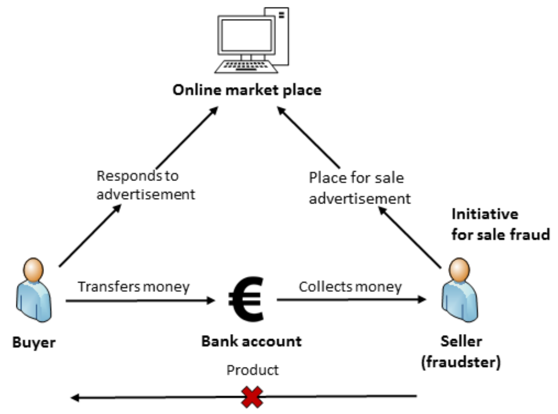


Figure 2: For sale fraud

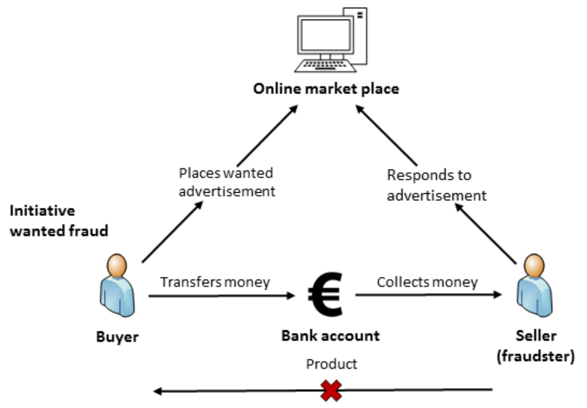


Figure 3: Wanted fraud

4.3 The process of handling a complaint

The process starts when a complaint has been filed, for which a civilian has two options; either online at www.politie.nl or at a local police station. Between 150 and 200 online trade fraud complaints are filed each day (Inspectie Veiligheid en Justitie, 2015). Due to the policy of the Dutch police to refer online trade fraud victims to the website, close to 90% of all complaints are filed online.

After a complaint has been filed online it will first be stored in a demilitarized zone (i.e. secured environment) hosted by KPN. A notice will be sent to the Dutch banks and commerce sites containing the suspect's username and account number. Besides this notice, both LMIO and DLOC will receive an e-mail containing the full complaint. At DLOC, an administrative police unit that handles incoming data, employees will enter the complaint into the enforcement database, BVH, to be accessible by all police units, which will further be explained in Chapter 4.3.1. Since LMIO does not make use of the BVH, but uses a stand-alone system, a complaint is also registered by LMIO in their database. To ensure that LMIO's database is up-to-date with the BVH, e-mails are sent by DLOC to notify LMIO of changes to a complaint (Peters, 2016b).

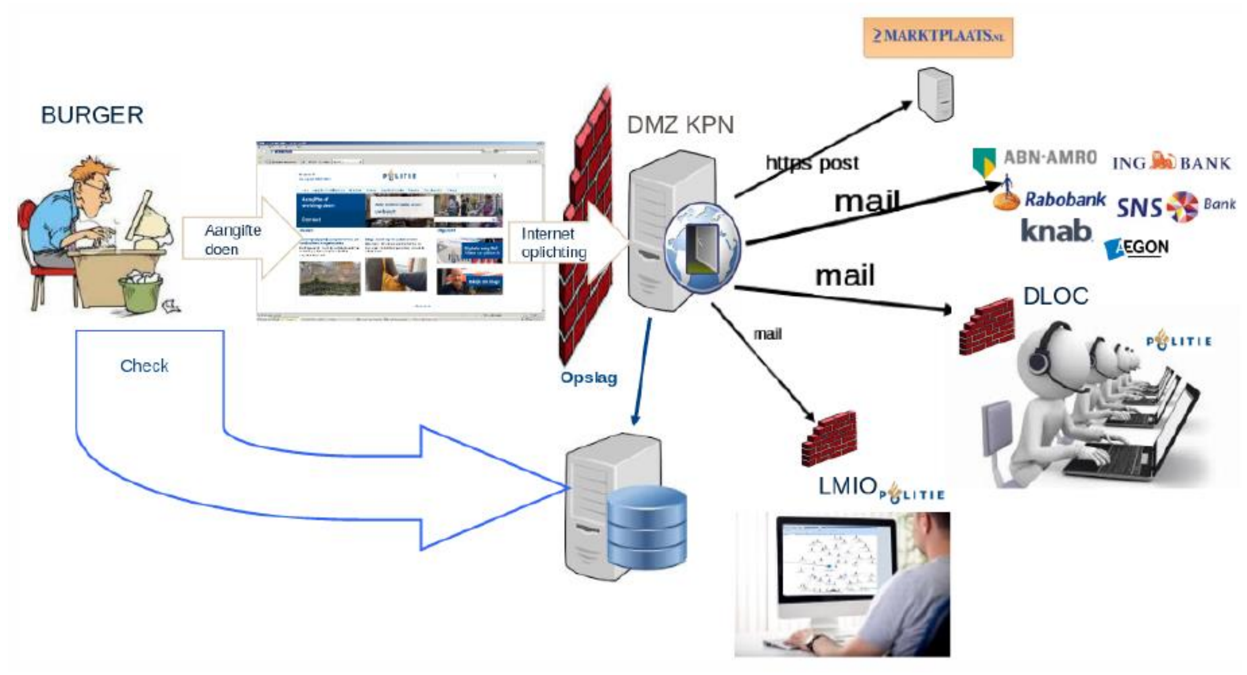


Figure 4: Information flow of a complaint[9]

An overview of the dataflow is included in Figure 4. Complaints that have been filed online are not official criminal complaints in the legal sense, since they are not signed. However, LMIO treats them as legitimate complaints and requests for an autograph when they are utilized in a case (Inspectie Veiligheid en Justitie, 2015).

When filing a complaint at www.politie.nl the complainant is asked to enter his own personal details as well as the personal details of the other party insofar they are known. Next to this, the online web form contains a free-text field which the complainant can use to tell the story that led to the complaint. An online trade fraud complaint thus consists of a story of what happened, combined with the identity of a suspect. In many cases though, a suspect tries to hide his true identity using a fake identity. With respect to chapter 3.2, where four types of cases were discerned, an online trade fraud case can thus either be a verification case or an investigation case, depending upon the (un)known identity of the suspect.

4.3.1 DLOC process of handling a complaint

An online trade fraud complaint received by DLOC is processed and entered in the BVH to be accessible by all police units. Employees at DLOC have three main responsibilities:

1. Ensuring the completeness of a complaint: when a complaint is found to be incomplete, employees at DLOC will ask follow-up questions to the complainant to complete the complaint.

[9] Source: Peters (2016b)

2. Classifying a complaint as civil or criminal: when a complaint is received by DLOC it is classified as either civil or criminal, since cases can only be built upon criminal complaints. A civil complaint is not registered in the BVH.
3. Processing complaint withdrawals: when a complaint is withdrawn online by the complainant this has to be manually processed in the BVH. Next to this an e-mail is sent to LMIO to inform them of the withdrawal of the complaint.

The three responsibilities are contained in two processes which are depicted in Figure 5 and Figure 6: registering and withdrawing a complaint, and summarized in Table 1 and Table 2. Both processes will be explained in detail below based upon interviews conducted at DLOC.

<u>Registering a complaint</u>

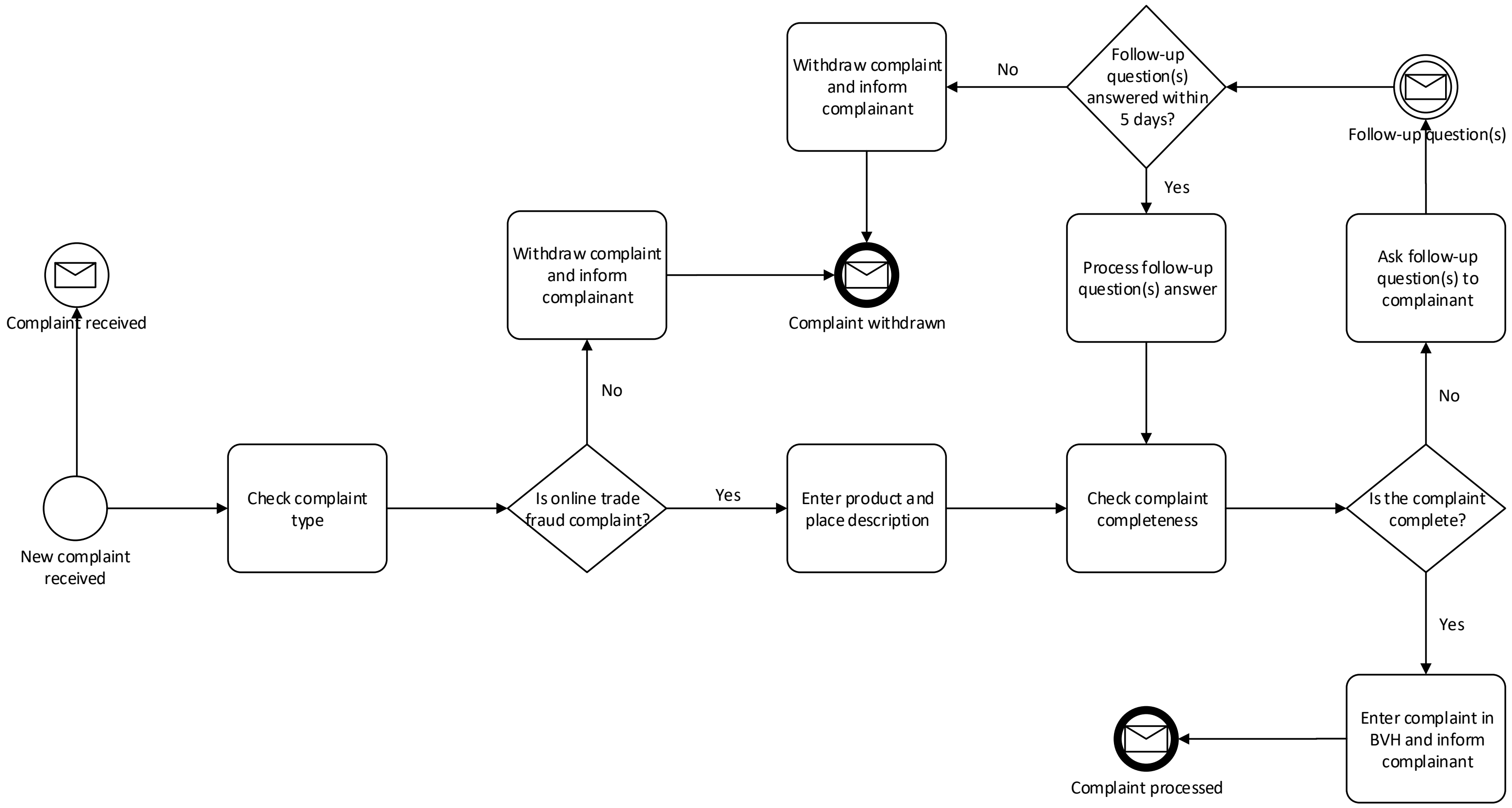

Complaint received
New complaint received
Check complaint type
Is online trade fraud complaint?
No
Yes
Withdraw complaint and inform complainant
Complaint withdrawn
Withdraw complaint and inform complainant
No
Follow-up question(s) answered within 5 days?
Yes
Follow-up question(s)
Enter product and place description
Process follow-up question(s) answer
Check complaint completeness
Is the complaint complete?
No
Ask follow-up question(s) to complainant
Yes
Enter complaint in BVH and inform complainant
Complaint processed

Figure 5: DLOC process of registering a complaint

The process of registering a complaint will be explained using the example of a complainant buying train tickets on Marktplaats but not receiving them. After the complaint has been filed online, an initial confirmation e-mail will be send to the complainant containing the reference number required for communication. The complaint will then be added to the complaint module which is used by employees at DLOC for handling a complaint. To minimize the time spent on complaints that cannot be used for building a case, a complaint will first be judged for its applicability (e.g. civil/criminal). Since, in our example, the complainant did not receive any goods which were ordered online, the complaint should be considered as online and criminal and thus as a relevant complaint. An irrelevant complaint will be marked as withdrawn in the complaint module and both the complainant and LMIO will be notified. For a relevant complaint, a process begins in which the complaint is refined according to police standards.

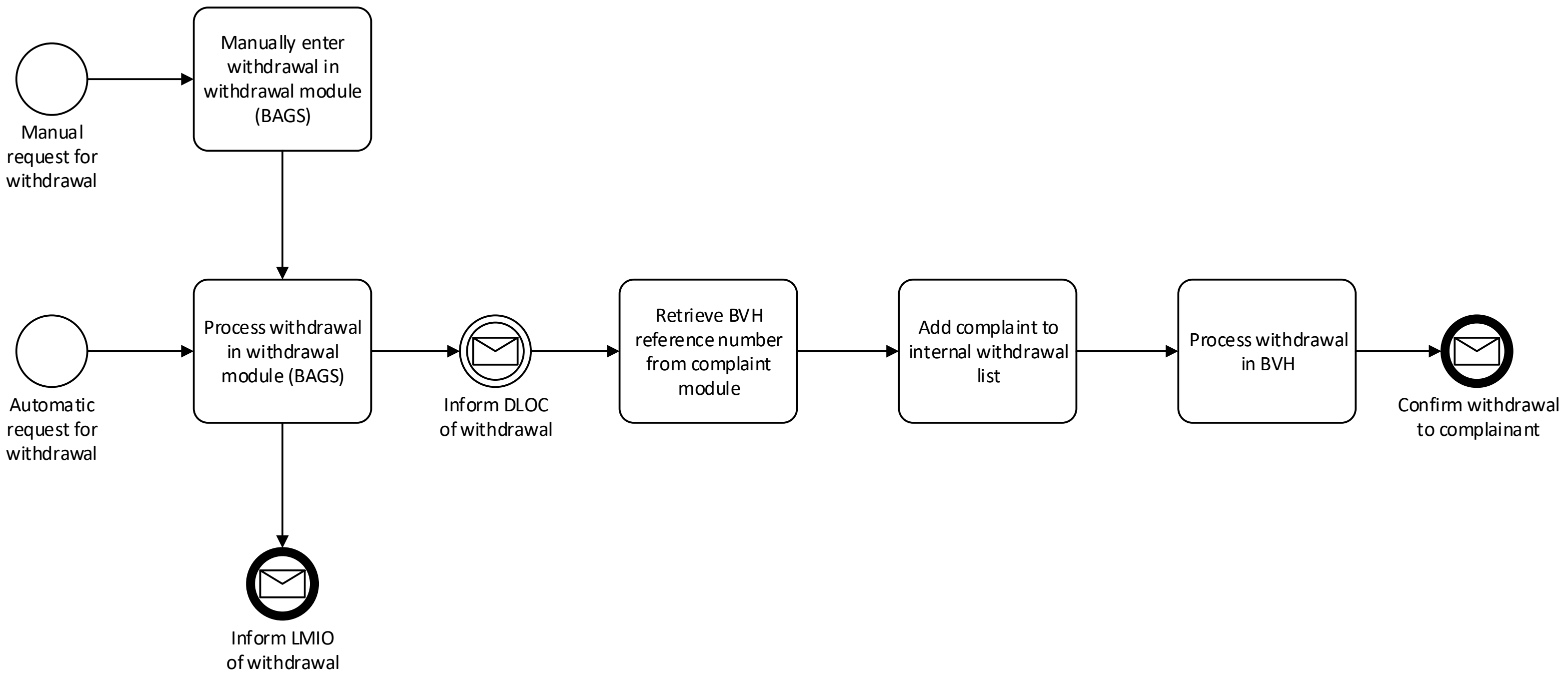
First, a product category (e.g. NS Tickets) has to be selected based upon the description given by the complainant. It may occur that a street name or place description has not been adopted correctly in the complaint module due to a misunderstood article or period (e.g. Prof. Dr. Koslaan as Profdr Koslaan), in which case it has to be manually adjusted. Next, a complaint is judged for its completeness to ensure it can be used by LMIO when building a case. If a complaint is found to be incomplete, one or multiple follow-up question(s) will be asked to the complainant, which have to be answered within 5 days, otherwise a complaint will be withdrawn. Follow-up questions will continue to be asked until the complaint is judged as complete. The complaint will then be entered in the BVH and a confirmation e-mail will be send to the complainant containing a new reference number. All tasks involved in registering a complaint have been summarized in Table 1.

Table 1: Tasks involved in registering a complaint

Nr.	Task	Input	Output	System
1	Check complaint type	A single complaint which has been filed through www.politie.nl	The type of the complaint	Complaint module
2	Withdraw complaint and inform complainant	A complaint which cannot be classified as an online trade fraud complaint or has not been responded to within time	A withdrawal e-mail send to both the complainant and LMIO	Complaint module
3	Enter product and place description	An unprocessed complaint	The complaint complemented with product and place description	Complaint module
4	Check complaint completeness	A complaint complemented with product and place description	An overview of the completeness of the complaint	Complaint module
5	Ask follow-up question(s) to complainant	An incomplete complaint	An e-mail send to the complainant containing follow-up question(s)	Complaint module

6	Process follow-up question(s) answer	An e-mail by the complainant containing the answer(s) to the follow-up question(s)	The complaint complemented with the answer(s) to the follow-up question(s)	Complaint module
7	Enter complaint in BVH and inform complainant	A complete complaint	The complaint entered in the BVH and a confirmation e-mail send to the complainant and LMIO	BVH

Withdrawing a complaint



Manual request for withdrawal
Manually enter withdrawal in withdrawal module (BAGS)
Automatic request for withdrawal
Process withdrawal in withdrawal module (BAGS)
Inform DLOC of withdrawal
Retrieve BVH reference number from complaint module
Add complaint to internal withdrawal list
Process withdrawal in BVH
Confirm withdrawal to complainant
Inform LMIO of withdrawal

Figure 6: DLOC process of withdrawing a complaint

It often occurs that a complainant decides to withdraw his complaint, for example when an ordered product is received after filing the complaint. Employees at DLOC receive between 25-30 withdrawal requests per day (i.e. 15-20% of the daily amount of complaints), which have to be processed in the BVH. Ideally, a complainant withdraws his complaint using a link provided in the confirmation e-mail sent when filing the complaint. This link is connected to the withdrawal module, BAGS, which automatically informs both LMIO and DLOC of the withdrawal. However, 10-20% of all withdrawal requests are received through other sources (e.g. the general contact form on www.politie.nl). These requests first have to be manually entered into the withdrawal module by employees at DLOC before the process of handling a withdrawal can be continued.

A BAGS generated e-mail contains the complaint module reference number and a reason for withdrawal. Since the complaint module reference number cannot be used in the BVH, the

BVH reference number first has to be retrieved from the complaint module. Next, both numbers will be added to an excel file containing references to all complaints which have to be withdrawn. Finally, a withdrawal is processed in the BVH and a withdrawal e-mail is send to both the complainant and LMIO. All tasks involved in withdrawing a complaint have been summarized in Table 2.

Table 2: Tasks involved in withdrawing a complaint

Nr.	Task	Input	Output	System
1	Manually enter withdrawal in withdrawal module (BAGS)	An e-mail by the complainant containing a request for withdrawal	A request in the withdrawal module to withdraw the complaint	BAGS
2	Process withdrawal in withdrawal module (BAGS)	An automated request for withdrawal by the complainant	A withdrawal e-mail send to LMIO and DLOC	BAGS
3	Retrieve BVH reference number from complaint module	A withdrawal e-mail containing a reference number to a complaint	The BVH reference number matching the complaint module reference number	Complaint module
4	Add complaint to internal withdrawal list	A text file containing the complaint module and BVH reference number	Excel file containing an additional row for the complaint	Excel
5	Process withdrawal in BVH	An excel file containing the complaint to be withdrawn	The complaint edited in the BVH and a withdrawal e-mail send to the complainant	BVH

4.3.2 LMIO process of handling a case

When a complaint has been received by LMIO, it is handled according to a standard process (Inspectie Veiligheid en Justitie, 2015), which has been depicted in Figure 7. Each step of the process will be explained in more detail and connected to the generic process of solving a case as seen in Chapter 3.

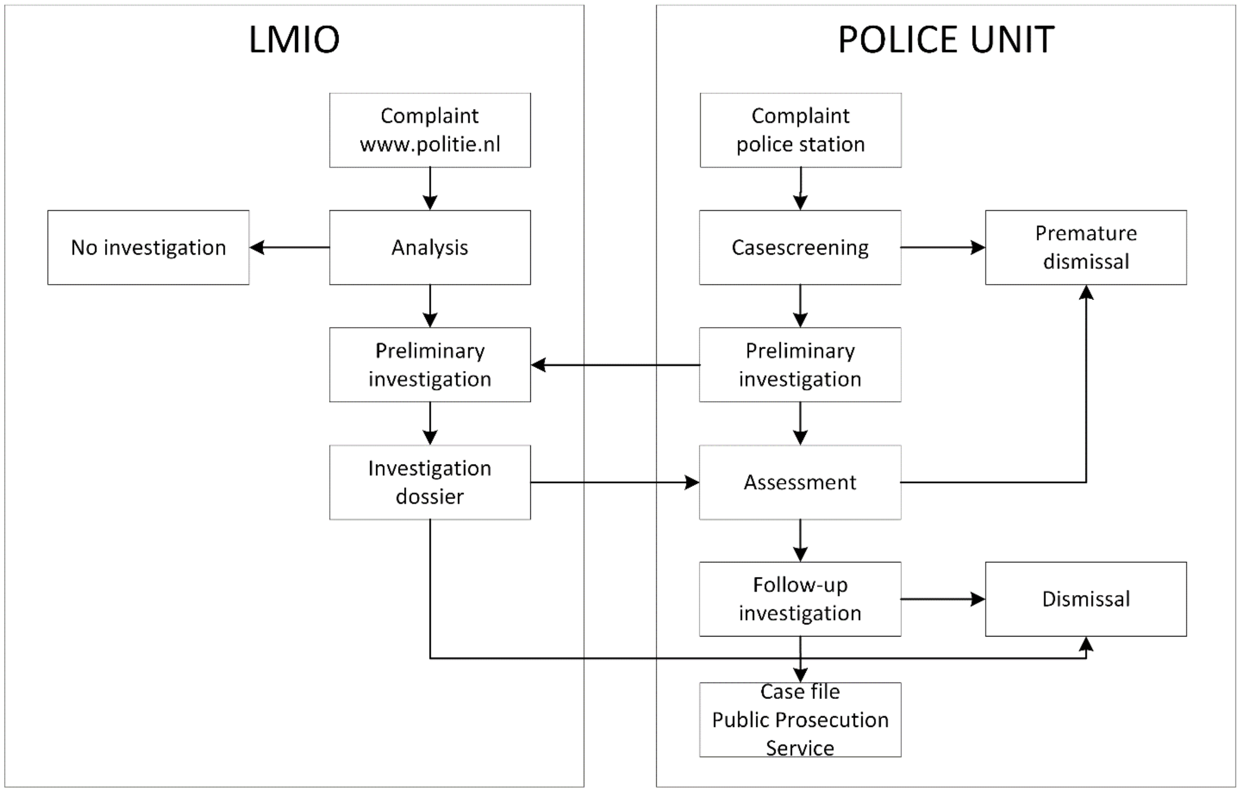


Figure 7: LMIO process of handling a case[10]

<u>*Analysis*</u>

LMIO receives between 150 and 200 complaints per day (Inspectie Veiligheid en Justitie, 2015). These complaints have to be analyzed to determine which could lead to an interesting case. For this, LMIO's analysts make use of iBase, which is a relational database used for recording and analyzing data. Each day, the complaints - both civil and criminal - are imported into iBase and cleaned; empty entities and unrealistic data, such as phone number 0123456789, are removed. Using iBase, entities are grouped based upon features in the data (e.g. similar account numbers or e-mail addresses) and connections between entities are discovered. Once a month, a list of suspects' account numbers is extracted from iBase and sent to the police units, which may help them in their own cases.

[10] Source: Inspectie Veiligheid en Justitie (2015)

Next to managing data and creating overviews, LMIO also investigates complaints. As was mentioned in chapter 3.2, not every possible case the police encounters is investigated; a pre-selection has to be made by LMIO's analysts to decide on which complaints should be investigated and which should not. For this pre-selection, four criteria are used:

1. Are there questions surrounding the entity elsewhere in the police organization?
2. Has the entity committed more frauds?
3. Is the suspect a minor?
4. How high are the costs of the damage?

Most investigations result from criteria 1, where individual police units request for information surrounding a specific entity, leaving LMIO less time to perform their own investigations on interesting entities. To determine which complaints the remaining time should be spent on, the analysts combine criteria 2-4. An investigation is always started if the suspect is a minor, has committed more than 10 frauds, or if the costs of damage exceed 5.000 euro. In case of more remaining time, intuition is used for selecting complaints worth investigating (Inspectie Veiligheid en Justitie, 2015).

If a complaint is further investigated, a dossier is constructed by the analysts consisting of [1] a detailed graph of all other entities connected to the entity, [2] all transactions of the entity, [3] the full complaint, and [4] a summary of each complaint connected to the entity. This dossier is then sent to the police unit of the city in which the suspect resides. The respective police unit can determine to preliminary investigate the possible case on their own, or have LMIO preliminary investigate the case.

Preliminary investigation (dossier)

If a dossier is handed over to LMIO for preliminary investigation, a police planner will be assigned to it. This police planner will, based upon the constructed dossier, try to recover the suspect's identity and determine whether a crime has been committed with respect to article 326 SR. To achieve this, requests for information are set out to relevant third parties (e.g. banks, commerce sites) for additional information regarding entities in the dossier, for example the identity of the holder of an account number. LMIO's police planners, as opposed to the analysts, do not have a standard approach for handling a complaint, due to the unknown availability of information. However, following the top-down approach mentioned in chapter 3.2, a police planner builds a case, using the story as the center of the investigation (de Poot et al., 2004).

During the investigation, obtained information is stored in Summ-IT, a nation-wide registration system to support the processes of an investigation. When a case has been fully investigated, this system is used as input for the case file which will be sent to the police unit. In this case file the crime will be described according to a predefined template, which is also included in Summ-IT. For constructing this case file, the police planners follow four general steps in an undefined order:

- Read and describe complaints;
- Report and describe investigatory findings;
- Create a general story according to the Golden W's, as mentioned in chapter 3.1;
- Apply article 326 of the Dutch Criminal Code to the case and describe the committed prosecutable facts.

After the case file has been created, it will be included in the dossier which will be physically sent to the police unit of the city in which the suspect resides. This dossier consists, besides the case file, of a project proposal, a handover document, a weighting document, a case analysis, and all documents used in building the case. The respective police unit will assess the quality of the dossier and determine based upon both this assessment and the availability of human resources whether a follow-up investigation will be conducted to ultimately present a case file to the Public Prosecution Service (Inspectie Veiligheid en Justitie, 2015).

5 Applying Data Mining at DLOC/LMIO

Before machine learning techniques can be used to (partly) automate the complaint handling process by DLOC and LMIO, it first had to be assessed where and how this should be applied. For this, interviews have been conducted at both DLOC and LMIO which were combined with the workflows as explained in chapter 4.3.

After a complaint has been filed by the complainant, it will immediately be sent to both DLOC and LMIO, who will work concurrently on the same complaint and communicate alterations to a complaint by e-mail. A potential solution which can be used by both DLOC and LMIO would ideally have to influence the process before a complaint is received. This implies influencing the process when filing a complaint, which is already being researched by Bex et al. (2016). Complementing the research by Bex et al., it has been opted to focus in this research on (partly) automating the process of either DLOC or LMIO. Even though this focus could restrict this research from being directly influential to both DLOC and LMIO, it does allow for indirect effects on the other process, due to the e-mail communication about the complaints. If, for example, the process at DLOC is enhanced, alterations can be faster communicated, thereby reducing the time spent by LMIO on complaints not worth investigating and improving the overall process of handling an online trade fraud complaint.

In the interviews it became obvious that a single complaint flows through a set of general steps at LMIO, which is depicted in Figure 7. However, since individual complaints differ substantially from one another, a tailored approach is required when analyzing and investigating a complaint, which makes it difficult to directly influence LMIO's process of handling a complaint. At DLOC, the processes of registering and withdrawing a complaint are not influenced by the details of a complaint, which enables the possibility to (partly) automate a process. Interviews at DLOC made it apparent that the most time intensive task was the answering of complainant's questions received via e-mail, followed by the registering of a complaint, and finally the withdrawing of a complaint. Regarding the problem solving urgency the same ranking was appointed.

Since the task of answering complainant's questions was designated as the most urgent problem, it was first explored how this could be improved using machine learning techniques. A possible solution could be a classifier which evaluates the content of an e-mail to predict the type of question, based on which a reply e-mail is formulated. However, since no dataset is available containing received e-mails, it was not possible to automate the answering of complainant's questions. Next, the process of registering a complaint, as illustrated in Figure 5, was explored for its possibilities regarding (partly) automation. Since through the research of Bex et al. (2016) the information contained in a "raw" complaint will be significantly improved in the nearby future, the cycle in which follow-up questions are asked and processed will become redundant. Therefore it has little added value to focus on automating this section of the complaint registering process. This leaves two tasks remaining in the process to be (partly) automated, namely the checking of the complaint type and the completing of the product and place description. During the interviews it was mentioned that the completing of the product and place description was not time-intensive and should therefore not be given any attention. Automatically determining the complaint type however was described as a welcome feature, as was predicting whether a complaint will be withdrawn is not. Both predictions serve the same goal, namely the distinction between complaints worth investigating and those not worth. By categorizing complaints, more important complaints can be dealt with earlier, thereby increasing the processing speed of those complaints at DLOC. This also improves the process at LMIO, since feedback regarding a complaint withdrawal is received faster, which will reduce the time spent on complaints not worth investigating.

Based on the interviews and the dataset, it was opted to focus in this research on distinguishing complaints based on whether they will be withdrawn or not. In chapter 4.3.1 two types of withdrawal were explained: withdrawal by DLOC judgement or withdrawal by complainant request. Within these two types of withdrawal, many specific reasons exist, some of which are: "no internet fraud", "civil case", "double complaint", "ID fraud", "phishing", "money was recovered", and "goods delivered". A possible classification approach in this research could be the distinction of withdrawn complaints based on those reasons, however, since employees at DLOC are merely interested in whether a complaint will be withdrawn or not, it has been chosen to group all withdrawn complaints together. As a result, the classifier will thus categorize a complaint as either withdrawn or not withdrawn.

Categorizing complaints will occur once they are received in the Complaint module, which will alter the process as shown in Figure 5. Once a complaint has been received it will be positioned in either one of two bins: withdrawn or not withdrawn. Complaints positioned in the not withdrawn bin will continue to be processed as before, with an employee at DLOC judging its completeness and asking follow-up questions. However, for complaints categorized in the withdrawn bin, a multitude of new approaches could be followed. Potential approaches could be [1] automatically withdrawing a complaint once it is received, [2] once a week/month automatically withdrawing all complaints, or [3] manually processing the complaints at a later point in time. A desirable approach would rely upon the outcome of this research as well as DLOC's goal regarding complaint handling (e.g. focus on customer intimacy or process efficiency). If, for example, DLOC highly values a complainant to be treated in a correct manner and only wants complaints to be withdrawn with absolute certainty, complaints which have been classified as withdrawn with 50% certainty cannot be automatically processed. However, if process efficiency is important and only 15 complaints are categorized as withdrawn each day, a certainty of 50% could well be enough to automatically withdraw a complaint. Since utilizing the research results within DLOC depends upon many variables, of which some examples have been given, it must thus be decided after conducting this research, in cooperation with DLOC, how this will be done.

6 Data Mining in Crime Analysis

In this chapter, the principles of data mining and classification will be explained. It will be described how classification has been applied in crime analysis and discusses how it can be applied to online trade fraud complaints.

6.1 Describing data mining

Data mining can be viewed as a step in the Knowledge Discovery in Databases (KDD) process as described by Fayyad, Piatetsky-Shapiro, and Smyth (1996). They define KDD as "the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" (Fayyad et al., 1996, p. 30). In this process, data necessary to achieve an outcome is first selected, after which it is preprocessed and transformed to ensure the data is of sufficient quality and follows the required format for applying the desired data mining techniques. Examples of data preprocessing and transformation are the cleaning, generalizing, aggregating, discretizing, and reducing of the data. When the data has been prepared, data mining techniques can be applied to build models on a dataset. After a model has been created, knowledge can be extracted by applying it to a dataset. An example of such knowledge extraction could be a model that predicts the likelihood of someone having a heart disease based on a patient record to assist a general practitioner in his daily job. The created models in the data mining step are evaluated for their applicability and accuracy. Throughout the entire KDD process, it is possible to take a step back and perform alterations. An example for this could be that the previously prepared data does not comply with the standards of the selected data mining technique and has to be retransformed or that additional data is required for building accurate models. The KDD process as described by Fayyad et al. is depicted in Figure 8.

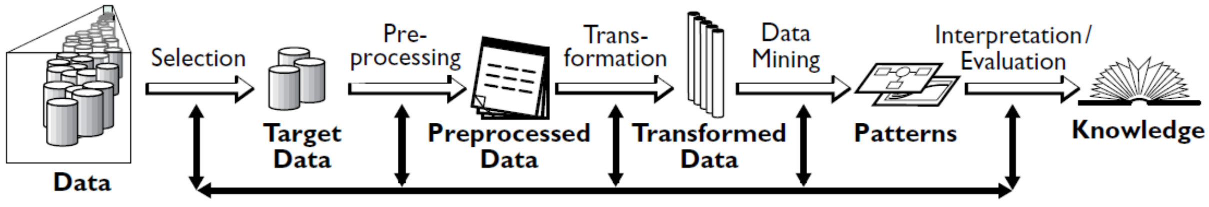


Figure 8: KDD process[11]

In this research, machine learning techniques will be utilized to build models in the data mining step of the process. Machine learning explores how computers can acquire new knowledge or enhance their performance based on data. Four types of machine learning techniques can be distinguished (Han, Kamber, & Pei, 2012):

- Supervised learning in which a training set with correctly labeled entities is used for training models on how to predict those labels and thereby classifying the data.
- Unsupervised learning in which a training set without labeled entities is used for training models on how to cluster the data based on its characteristics.
- Semi-supervised learning in which a training set with both labeled and unlabeled entities is used for training models on how to predict the class labels and cluster the data.
- Active learning in which a user actively evaluates intermediate results to improve the accuracy of built models.

6.2 Classifying data using machine learning

As stated in the main research question the purpose of this research is to (partly) automate online trade fraud complaint handling. For this machine learning techniques will be used which make use of the principle of classification. A classification approach involves the building of a classifier (or model) to predict class labels and consists of two phases; a training phase in which the classifier is built on a dataset with entities and their corresponding class labels, and a classification phase in which class labels for given entities are predicted using the classifier. The goal of a classifier is to correctly predict all class labels. To determine the accuracy (i.e. number of correct predictions) of a classifier it is tested on a dataset and the percentage of correctly classified entities is taken. If this dataset would be the same dataset as used for building the classifier, the accuracy would most probably be overly optimistic due to the nature of a classifier to overfit the data (i.e. tuning the classifier to specific entities which are not present in the general dataset). Therefore, in the training phase, a training set is used

[11] Source: Fayyad et al. (1996)

for training the classifier and a test set, which is independent of the training set, is used for determining its accuracy. Since, in this research, the entities in the training set have a corresponding class label, classification is a supervised learning technique (Han et al., 2012). Numerous machine learning techniques exist, of which the ones initially applicable to this research will be explained below. This list of techniques is based on the work of Peters (2016a) and a supporting literature study.

<u>Decision tree classification</u>

With decision tree classification, a tree structure is constructed from a dataset consisting of internal nodes, branches, and leaf nodes. Each internal node represents a condition for an attribute (e.g. age greater than 25), each branch the outcome to this condition (e.g. yes/no), and each leaf node a class label for this condition (e.g. applies for loan/does not apply for loan). Decision trees are a popular machine learning technique due to their intuitive nature, reliable accuracy and fast operational runtime. To decide the conditions on which a tree should be constructed, decision tree algorithms make use of attribute selection measures. Attribute selection measures are heuristics to determine the condition used for splitting which results in the best split on a given dataset. Popular decision tree induction algorithms are ID3, C4.5, and CART which make use of attributes selection measures such as the resubstitution error, Gini-index, and entropy (Han et al., 2012).

In 1995, Freund and Schapire (1995) proposed an algorithm, Adaptive Boosting, which uses majority voting to combine the weighted output of many small decision trees based on their predictive accuracy. Even though AdaBoost holds a risk for overfitting on the training data due to this weighting, it has been shown to outperform many other machine learning techniques including regular decision trees (Schapire and Singer, 2000; Sebastiani, 2002; Weiss et al., 1999)

<u>Naive Bayesian classification</u>

Naive Bayesian classifiers are part of the Bayesian classifiers which predict class membership probabilities based upon Bayes' theorem. A naive Bayesian classifier models a class distribution with conditional probabilities which can be used to predict a class label. With naive Bayesian classifiers, the attributes which influence a class label are expected to be independent of, and not of any influence to each other, which is known as class-conditional independence. Even though this assumption for independence does not always hold, naive Bayesian classifiers are popular due to their low complexity, are known to accurately predict class labels, and are comparable to or even outperform more complex machine learning techniques (Han et al., 2012).

Naive Bayesian classifiers can be split up into two generative models, known as multi-variate Bernoulli and multinomial naive Bayes. The difference between these models can be found in the probability definition. With multi-variate Bernoulli naive Bayes, the probability of an entity belonging to a specific class is calculated by multiplying the binary probability of all attributes for that entity, including that of non-occurring attributes. Multinomial naive Bayesian classifiers calculate the probability of an entity belonging to a specific class by multiplying the probability of each occurring attribute for a specific entity, excluding non-occurring attributes. If, for example, an attribute “Marktplaats” does not occur for a given entity, it will thus be included in the probability prediction for a multi-variate Bernoulli naive Bayesian classifier and excluded in a multinomial naive Bayesian classifier. Both naive Bayes models are used as standard naive Bayes in research even though they differ substantially (McCallum & Nigam, 1998).

<u>K-nearest-neighbor classification</u>

K-nearest-neighbor (KNN) classifiers are an example of lazy learners, which do not perform any calculations on training sets until it is required for a test set. When an entity in a test set is received, the classifier loops over the training set to determine which entities are the closest to it with respect to its attributes. The closeness to a neighbor is determined using a distance metric (e.g. Euclidian distance as used in this research). This distance metric can, however, not be applied to all types of attribute values, for instance categorical values. K-nearest-neighbor classifiers are known to be computationally slow, but in terms of accuracy comparable to Bayesian classifiers (Han et al., 2012).

<u>Support vector machine classification</u>

With support vector machine (SVM) classification a dataset is separated based upon its characteristics using a hyperplane (i.e. decision boundaries) as illustrated in Figure 9. For this, the dataset is transformed to a higher dimension using non-linear mapping. By applying a proper non-linear mapping to an adequately high dimension, a hyperplane can always be found which separates a dataset with two classes. SVM classifiers are computationally slow, but known to be less sensitive to overfitting and highly accurate (Han et al., 2012).

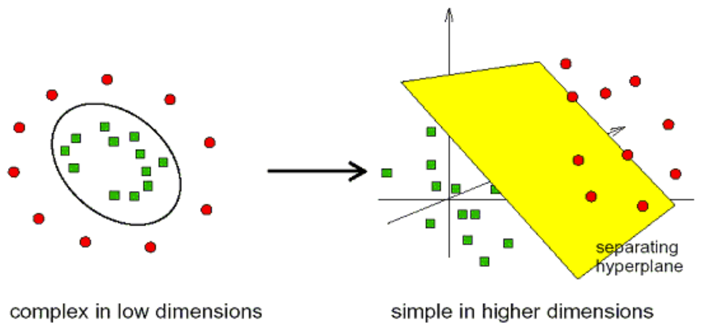


Figure 9: Support vector machine classification[12]

[12] Source: inovancetech.com

Logistic regression classification

Logistic regression classifiers model the probability that an entity belongs to a certain class. For this, a logistic function combining regression coefficients is used to transform the output of a linear function. Such a regression coefficient determines how much an attribute in the dataset influences the overall model. The regression coefficients required to predict the class of an entity are calculated in the training set and verified on the test set. Logistic regression classifiers are comparable to naive Bayesian classifiers in terms of accuracy (Witten & Frank, 2005).

Association rule classification

With association rule classification a set of IF-THEN rules is used to predict a class label. The IF part of a rule, the antecedent, can consist of multiple attributes, while the THEN section, the consequent, contains a single class prediction. An often occurring type of algorithm for learning this set of rules is the sequential covering algorithm (e.g. CN2 or RIPPER). Using this type of algorithm, a rule is learned one at a time using a greedy depth-first strategy, after which the entities covered by this rule are removed before a next rule is learned (Han et al., 2012).

Neural network classification

Neural network classifiers predict the class label of an entity using a structure of input/output units and weighted connections as depicted in Figure 10. A neural network is constructed over a predefined number of iterations, better known as epochs. In each epoch, the output of the last epoch is first entered in an input layer, which forwards it to a predefined number of hidden layers. A hidden layer contains a set of weighted units, which outputs are combined as input for a next unit. The weight of each unit is updated according to its predictive accuracy in the last epoch. After the data has gone through all hidden layers it is passed on to the output layer, where a class prediction can be made. A popular algorithm for computing a neural network is the backpropagation algorithm (Han et al., 2012).

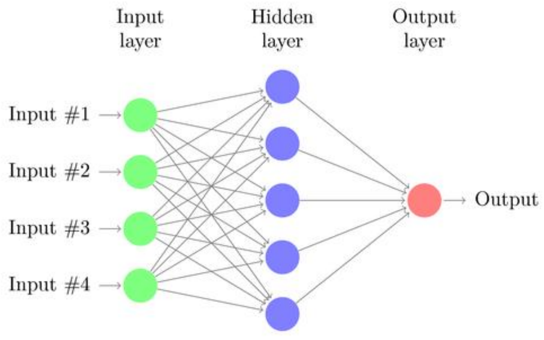


Figure 10: Neural network classification[13]

[13] Source: texample.net

Ensembling

To improve the classification accuracy and thereby fulfilling the goal of a classifier, it often occurs that the outcomes of individual classifiers are combined to create an overall classifier, which is known as ensembling (Han et al., 2012). For this, both similar machine learning techniques (i.e. bagging) as well as dissimilar machine learning techniques (i.e. stacking) can be combined. With bagging, a predefined number of training sets is sampled with replacement from the available dataset, and a set of classifiers using a similar machine learning technique is built based on these training sets. An overall classifier is created using the principle of majority voting, in which each individual classifier returns its outcome for a given test set and the outcome occurring most often is taken as the real outcome (Han et al., 2012). With stacking, this same principle applies, however, the individual classifiers can be built using different machine learning techniques (Wolpert, 1992). In general, ensemble methods are known to be more accurate than individual classifiers due to their robustness to overfitting (Han et al., 2012).

6.3 Classification in crime analysis

Data mining techniques, including classification, are increasingly being applied to the field of crime analysis. Chen et al. (2004) explored data mining techniques used for crime analysis, and Sharma and Panigrahi (2012) have categorized over 40 approaches using machine learning techniques for fraud detection.

Abdelhamid, Ayesh, and Thabtah (2014) have used associative classification, a technique in which both association rules and classification are combined, to accurately discriminate phishing websites from legitimate websites. Making use of Bayesian networks, Baumgartner, Ferrari, and Palermo (2008) have been able to predict the characteristics of a homicide offender based on crime scene variables (e.g. police report or autopsy report) more accurate than a team of police experts. These characteristics could be used by police officers to identify a possible suspect. Gupta et al. (2016) have compared naive Bayes, Bayesian network, decision tree, and association rule techniques for their speed and accuracy in classifying crimes and accidents in Denver City, and found that association rules result in the highest accuracy. In their collaboration with the West Midlands Police, Oatley and Ewart (2003) used a Bayesian network to predict whether a certain property in the UK's Midlands will be re-victimized or not and within what timespan. Next to this, a neural network was used to classify possible offenders for their likelihood of conducting unsolved crimes.

Some of the research performed in this field focuses mainly on how to deal with a skewed dataset (i.e. one class is overly present). In their research, Brause, Langsdorf, and Hepp (1999) combined a rule-based association system with a neural network to detect credit card fraud. Since credit card fraud cases are relatively rare compared to legal cases, Brause et al. dealt with a skewed dataset. For a classifier to be adequate on skewed data, its accuracy has to be higher than the percentage of the most occurring class in the dataset (e.g. the percentage of legal cases with respect to Brause et al.). Using their combined classifier, Brause et al. were able to achieve an accuracy of 99.955%. Phua, Alahakoon, and Lee (2004) have performed research on which classification method is best to be used for fraud detection with a skewed dataset. They proposed the use of a stacking-bagging method, in which a naive Bayes, neural network and decision tree classifier are combined.

6.4 Classifying online trade fraud complaints

Even though machine learning techniques have been used for many purposes in crime analysis, to the knowledge of the author only one research has focused on how to apply machine learning techniques to online trade fraud complaints (Peters, 2016a). The results of Peters are promising and show that machine learning techniques could be applied to online trade fraud complaints, however, the classification performance did not meet the required thresholds. In this research, the use of machine learning techniques will be increased and improved so that the required thresholds can hopefully be attained. Unfortunately, Peters only broadly described his followed procedures, making it hard to apply all of his findings to this research. Knowledge on how to classify online trade fraud complaints will therefore have to be a combination of the findings from Peters (2016a), an illation from literature on classification in crime analysis, and a deduction from literature on classifying textual and skewed data.

The dataset used in this research is an online trade fraud complaints dataset which has been provided by the Dutch National Police and has been preprocessed by Peters (Peters, 2016a). This dataset consists of 51.386 entries, which have all been manually labelled by employees at DLOC on whether a complaint has been withdrawn or not and for what reason. As explained in Chapter 4.3.1, a complaint can be withdrawn due to a complainant's request or DLOC's judgement. A complainant can request a withdrawal for any reason which is, when possible, categorized by employees at DLOC. When employees at DLOC withdraw a complaint on their own judgement this is done according to predefined reasons (e.g. civil, incomplete, no online trade fraud, double complaint). However, due to the large amount of withdrawal reasons, it cannot be deduced which withdrawals result from LMIO's judgement and which from a complainant's request.

In total, 8.609 (16.7%) entries have been labelled as withdrawn and 1.136 (2.2%) as civil, which is a reason for withdrawal and therefore a subset of the withdrawn entries. Due to the offset ratio of minority to majority the online trade fraud complaints dataset is to be considered as a skewed dataset. The dataset contains a total of 60 attributes, including the binary class labels, and contains a free-text field in which the complainant's story that led to the complaint is included. An anonymized and translated example of the free text field of a withdrawn and not withdrawn complaint is included in Table 3.

Table 3: Data example

Withdrawn	John Doe advertised a rental home. In hindsight it all appears to be fake.
Not withdrawn	I have bought a bottle of Dom Perignon and a bottle of Crystal 1999 from John Doe via Marktplaats and transferred 100 euro to NL01ABCD0123456789. Up to now, I have not received anything and John Doe does not respond to my e-mails.

In this research textual attributes of a skewed dataset will be used. Therefore literature studies on classification for textual and skewed data have both been performed and outcomes are presented in the next sub chapters.

6.4.1 Classification for skewed data

Similar to Brause et al. (1999) and Phua et al. (2004), the online trade fraud complaints dataset used in this research contains a high degree of skewness. This high degree of skewness involves one class being overly present in the dataset. Since a skewed dataset consists of a majority and a minority class, a relatively high accuracy can be attained when classifying every entity to the majority class. Standard machine learning techniques are designed to pay more attention to majority classes, and thereby often perform poorly on a skewed dataset (Japkowicz and Stephen, 2002). In the literature, two abstract solutions are given for solving this problem: [1] a data-level approach, and [2] an algorithm-level approach (Japkowicz and Stephen, 2002; Phua et al., 2004; Sun et al., 2007; Sun, Kamel, & Wong, 2009; Tang et al., 2009).

Regarding the data-level approach, Japkowicz and Stephen (2002) have evaluated whether alterations to the training set can improve the accuracy of a classifier on a skewed dataset. For this they compared the accuracy of a classifier trained on an unaltered dataset to the same classifier trained on a dataset altered using one of the following two techniques:

- Undersampling in which the entities in the majority class will be, at random, eliminated until a desired distribution is achieved.
- Oversampling in which the entities in the minority class will be, at random, duplicated until a desired distribution is achieved.

As a mixed approach, Japkowicz and Stephen also used cost-modification in which the costs of misclassifying a class depends upon the class distribution. If, for instance, the class distribution is 1:4, then the costs of misclassifying a minority entity will be four times as high as misclassifying a majority entity.

Japkowicz and Stephen found that a greater class imbalance, a greater dataset complexity (i.e. the amount of dimensions required for separating the dataset), and a smaller training set size all negatively influence the performance of a classifier. Regarding decision trees, Japkowicz and Stephen found that the accuracy of decision trees on linearly separable classes (i.e. classes which can be separated in two sides) is barely influenced by any class imbalances, however, for more complex separations (i.e. non-binary classes) decision trees were found to be most influenced by class imbalances followed by neural networks. Support vector machines did not show to be affected by any class imbalances. Regarding the data alteration technique, Japkowicz and Stephen discovered that both oversampling and cost-modification improve the performance of classification trees on a skewed dataset, however, undersampling has no effect in their research domain. For neural networks, under- and oversampling were found to both improve the performance of a classifier. Data alteration techniques, however, did not have any effect on the performance of support vector machine classifiers.

The three alteration techniques evaluated by Japkowicz and Stephen (2002) are also supported by Sun et al. (2007;2009) and Tang et al. (2009) as the best techniques. Phua et al. (2004) have used these techniques to evaluate the best data distribution for an automobile insurance fraud dataset. They found that either a 40:60 or a 50:50 distribution provides the most accurate classification results.

In this research, two of the techniques mentioned by Japkowicz and Stephen (2002) will be applied for redistributing the dataset, undersampling and oversampling. When undersampling the dataset, majority cases will be, at random, eliminated until a desired distribution is achieved. For oversampling the dataset, the SMOTE algorithm as proposed by Chawla et al. (2002) will be used. With SMOTE, a minority case is duplicated by randomly selecting one of its 5 nearest minority neighbors. Next, a hyperplane is created between the minority case to be duplicated and the selected minority case. Finally, the new minority case is created by randomly selecting a point on this hyperplane and joining the two minority cases for this point. SMOTE has been shown to be a reliable and robust redistribution technique in comparison to random oversampling and other redistribution techniques (Chawla et al., 2002; Liu, 2004).

Regarding the algorithm-level approach, ensembling as described in chapter 6.2 is mentioned in the literature as a method for improving classifier accuracy (Han et al., 2012; Sun et al., 2007; Sun et al., 2009; Tang et al., 2009;). Many ensembling methods have been proposed in the literature which all rely upon either bagging or stacking (Gao et al., 2008; Han et al., 2012; Tang et al., 2009; Wang, Zhang, & Wang, 2009). However, results obtained through using these ensembling methods heavily depend upon the type of data and, due to often conflicting results, no general theory can be deduced on how to apply them to the online trade fraud dataset. This means that the effect of the ensembling methods will have to be tailored to our dataset.

6.4.2 Classification for textual data

In his paper, Sebastiani (2002) presents an overview of machine learning techniques used for classifying textual data. By comparing the classification results of classifiers used in individual papers on the Reuters dataset, which is a multi-labelled collection of documents with a highly skewed distribution, Sebastiani is able to derive conclusions on classifier performance. Ensembling based (e.g. AdaBoost), SVM, logistic regression, association rule (e.g. RIPPER), KNN, decision tree, and neural network classifiers all have a high performance on textual data. Probabilistic (e.g. naive Bayes) classifiers, however, show a lower performance on classifying textual data. In the remainder of this sub chapter, individual research performed on textual classification will be presented.

Schapire and Singer (2000) designed an alternative version of AdaBoost, BoosTexter, and compared its classification results with KNN, naive Bayes, and RIPPER. The results of their evaluation showed that BoosTexter significantly outperforms the other classifiers. Research by Weiss et al. (1999) also showed that ensembling based decision trees outperform other machine learning techniques. In his paper, Joachims (1998) concludes that the performance of an SVM classifier is better than that of naive Bayes, decision tree and KNN classifiers. This finding is supported by Dumais et al. (1998) in their research on SVM performance for text classification. Yang (1998) compared 10 classifiers on their performance and discovered that KNN, logistic regression, and neural networks outperform the other classifiers. In his research, however, Yang did not include a SVM classifier. In a follow-up research, Yang and Liu (1999) included a SVM classifier in their analysis and concluded that it is similar in performance to KNN and logistic regression. This study, however, showed that neural networks underperform compared to these three classifiers.

By combining the results from all literature studies it can be concluded that the machine learning techniques described by Sebastiani (2002) cover the spectrum of techniques used for classifying textual, skewed, and criminal data. The performance of the mentioned machine learning techniques will therefore be analyzed in the data analysis.

6.4.2.1 Input for textual classification

One common feature between each classifier for textual data is the input used for training and testing it. Such input consists of a collection of documents, known as a corpus, and a corresponding class label. In this research, a document is represented by a single complaint. Since a corpus, however, is merely a collection of (transposed) documents and holds no information on itself, it has to be converted into a term-document matrix. In a term-document matrix, the frequency of every term occurring in each document (i.e. complaint) is contained. For this research, it has been opted to use terms ranging from 1 to 3 adjacent words (i.e. 1:3-grams), which have been tokenized using the NLP suite Frog (Van den Bosch et al., 2007). Alternative features could also be used for creating the terms such as part-of-speech, in which words are assigned to a category based on their use and function. By combining the 1:3-grams created for all individual complaints, the set of terms for the term-document matrix is obtained. As with classification for skewed data, alterations on a data-level approach can be applied to the term-document matrix. A distinction is made between word weighting techniques and word normalization techniques.

Word weighting techniques transpose the frequencies of each term in the term-document matrix based on the estimated importance of a word in a document. In this research, three word weighting techniques will be compared with respect to their influence on a classifier's performance: no weighting, binary weighting, and term-frequency/inversed-document-frequency (TF-IDF) weighting. By applying no weighting (i.e. not transposing the frequencies), a baseline can be created to which the performance of the two other weighting techniques can be compared. With binary weighting, the frequency of all occurring terms is transposed to 1, so that a term can either be occurring or not occurring. Research has shown that binary weighting outperforms other weighting techniques on a similar dataset (Pang et al. 2002; Schneider, 2004). TF-IDF is one of the most common weighting techniques in Information Retrieval and is increasingly being used for text classification (Zhang et al., 2011). By multiplying the frequency of a word with the inverse of the amount of documents in which it exists, TF-IDF assigns a weighting based on a word's relative value. For example, the Dutch article *"de"* is used very often and has little predictive value resulting in a low weighting, while the verb *"oplichten"* is more predictive and therefore receives a higher weighting.

Word normalization techniques transpose a word into its standard form and are often applied in the field of NLP to reduce the dimensionality of a corpus. Two specific word normalization techniques will be applied in this research, stemming and lemmatization. With stemming, a word is reduced to its standard form (i.e. stem) by removing the suffix. In this research the stemming algorithm as proposed by Porter (1980) will be used, which has been adapted for the Dutch language. Lemmatizers utilize lexica to transpose a word to its canonical form (i.e. lemma) depending upon its morphology. For example, the Dutch word *"bakken"* used as a noun has the lemma *"bak",* while the verb has the lemma *"bakken"*. In this research the NLP suite Frog will be used to obtain the lemmas of all words in the corpus (Van den Bosch et al., 2007). The difference between stemming and lemmatization can be clearly exemplified using the Dutch word *"zochten"*. The stem for this word is *"zocht"* while the lemma is *"zoeken"*.

6.4.3 Evaluating classifiers for (non-)textual skewed data

Since with a skewed dataset a relatively high accuracy can be attained when classifying every entity to the majority class, the accuracy of a classifier is no reliable measure for how well it performs. Instead of looking at the accuracy, it is proposed to make use of the true/false positive/negative rates, as summarized in Table 4 (Sun et al., 2007; Han et al., 2012).

Table 4: True/false positive/negative rates

Actual class \ Predicted class	A	B
A	TP	FN
B	FP	TN

True Positive Rate: $TP_{rate} = \frac{TP}{TP + FN}$ [6.1]

True Negative Rate: $TN_{rate} = \frac{TN}{TN + FP}$ [6.2]

False Positive Rate: $FP_{rate} = \frac{FP}{TN + FP}$ [6.3]

False Negative Rate: $FN_{rate} = \frac{FN}{TP + FN}$ [6.4]

An evaluation measure often used with textual data for analyzing the performance of classifying the positive class is the F_{β}-measure, from here on called F-measure (Equation 6.7). This measure makes use of both the recall (i.e. TP_{rate}), which shows how many of class A's entities have been correctly classified, and the precision, which shows how many entities were correctly classified as class A. In general, a beta of 1 is used for the F-measure, as will be in this research, which makes it a harmonic mean between the recall and precision. It applies the same weight to both the recall and precision, thus a high F-measure ensures that most of the relevant entities have been correctly classified (Sun et al., 2007; Han et al., 2012). The equations for the precision, recall, and F-measure are given in equation 6.5, 6.6, and 6.7.

When calculating the average F-measure in a binary classification problem two approaches can be followed: micro-averaging or macro-averaging. With micro-averaging, the respective class weights are taken into account when determining the overall F-measure of two classes, while macro-averaging uses the arithmetic mean (Sokolova and Lapalme, 2009; Yang and Liu, 1999). The equations for the micro- and macro averaged F-measure are given in equation 6.8 and 6.9.

$$\text{Precision} = \frac{TP}{TP + FP} \quad [6.5]$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad [6.6]$$

$$F_{\beta}\text{-measure} = \frac{(1 + \beta^2) * precision * recall}{(\beta^2 * precision) + recall} \quad [6.7]$$

$$F_{micro} = \frac{(n * F_{classA}) + (m * F_{classB})}{(n + m)} \quad [6.8]$$

$$F_{macro} = \frac{F_{classA} + F_{classB}}{2} \quad [6.9]$$

The application of the F-measure for analyzing the performance of classifying skewed data is shown in the following example using the confusion matrix in Table 5. Here, 1000 instances of class A and 100 instances of class B are used. Of the 1000 instances in class A, 800 have been correctly classified, where for class B, 20 instances have been correctly classified. Overall, an accuracy of $820 / 1100 = 0.745$ is achieved on the test set. However, the F-measure of class B is only 0.125, which is significantly lower than the accuracy. Furthermore, with an F-measure for class A of 0.851, the micro-averaged F-measure is 0.785, while the macro-averaged F-measure is 0.488. Due to the skewed nature of the test set, a fairly high accuracy and micro-averaged F-measure can be achieved by correctly classifying a large part of the majority class, while the minority class is undermined. Since in (skewed) text classification, it is generally important to correctly classify all classes instead of correctly classifying a single class, it is proposed to use the macro-averaged F-measure as a performance evaluation measure (Forman, 2003; Yang and Liu, 1999).

Table 5: Example confusion matrix

		Predicted class	
		A	B
Actual class	A	800	200
	B	80	20

7 Results

In this chapter, the results of the data analysis will be described. For each classification technique, an overview will be given of the difference in performance between the resampling, word normalization and word weighting techniques. Furthermore, important decisions made in this research as well as the framework used for conducting the experiments will be explained in detail.

7.1 Research considerations

Data acquired through research are mere numbers if it has not carefully been excogitated how to obtain it and what to do with it. For obtaining the data, decisions regarding resampling, word weighting, and word normalization have already been explained in Chapter 6.4 based on performed literature studies. However, considerations for creating the research framework and evaluating the data resulting from this framework have not yet been discussed and will be in this sub chapter.

7.1.1 Framework creation

Even though most decisions for creating the research framework are induced by earlier decisions regarding the structure of this research, two decisions remain: [1] whether and how to select features for building a classifier, and [2] how to obtain the most accurate results.

When building a term-document matrix on the training set, more than 2.000.000 terms are generated consisting of 1:3-grams. Since most classification algorithms cannot handle such a vast amount of terms, a set of terms has to be selected which will be used for training the classifier. To help in the decision on which terms to select, feature selection algorithms can be used which try to generate the best classification results using the least amount of features. Various feature selection algorithms exist (e.g. Information Gain, Odds Ratio) which have all extensively been evaluated with respect to text classification (Forman, 2003; Yang & Pedersen, 1997). Even though feature selection algorithms are found to often positively influence classification results, it has been opted to use the most occurring terms over a standard feature selection algorithm. Underlying this is the focus of this research, which is on determining the influence of resampling, word weighting, and word normalization techniques on classifier performance. When applying feature selection, another dimension would be added to the problem, which would make it hard discriminate the influence of the above mentioned data alteration techniques. Through a small experimental setup it was found that the classification performance improved when increasing the amount of features used for training up to 100 after which it stabilized. To reduce the runtime of the research framework, while still obtaining the best possible results, the amount of features used for training the classifiers was chosen to be fixed at 100. Selecting the 100 most occurring terms makes the classification algorithm unbiased in the distinction of the minority and majority class, thereby explicitly showing the influence of the data alteration techniques to that process.

Next to being discriminable, results that have been obtained should also be an accurate demonstration of the truth and not a positive or negative outlier. Such positive or negative outliers could be the result of a small dataset where not enough data is available to accurately train and test a classifier (Beleites et al., 2013). To prevent outlier results, it is proposed to make use of 10-fold stratified cross validation, which reduces the variance of the outcome (Kohavi, 1995). With 10-fold stratified cross validation, the dataset is split in 10 folds of similar size, which all hold the same class distribution ratio as the original dataset. When training and testing a classifier 9 of the folds are used for training, while the remaining fold is used for testing it. This process is repeated 10 times, so that each fold is used 9 times for training and once for testing. After all 10 iterations have been performed the mean of the individual results is taken as the overall result.

7.1.2 Result evaluation

When analyzing the classification performance of a classifier trained on skewed textual data, a macro-averaged F-measure is often used as was described in Chapter 6.4.3. In the context of this research, however, it has been assessed whether this is the best evaluation measure. Since the purpose of a classifier is to have the best performance with respect to a certain goal, it was essential to first determine this goal before choosing the evaluation measure. During the interviews conducted at DLOC, the goal was described to be the correct categorization of a complaint in either one of two bins "withdrawn" or "not withdrawn" based on whether a complaint will be withdrawn or not. Furthermore, no bin was emphasized as being more important than the other. Currently, it is unknown what will happen with complaints in the "withdrawn" bin, as was explained in Chapter 5. This does, however, certainly influence the choice for a classification metric. If the certainty for a complaint to be placed in this bin is large, the precision should be as high as possible, opposed to a large recall when it is important that as many to be withdrawn cases are covered. Even though no bin was emphasized as being more important, it did make sense during the interviews at DLOC that, independent of the utilization, the "withdrawn" bin should contain as little incorrect complaints as possible. For the purpose of this research and choosing an appropriate classification metric, it has therefore been decided that the precision of the "withdrawn" class (i.e. minority class) should be as high as possible.

In Chapter 6.4.3, the precision was given as the True Positives divided by both the True Positives and False Positives. The precision with respect to the minority class can be calculated using equation 7.3. From this equation it can be derived that the precision can increase if either more minority classes are correctly classified or less majority classes are incorrectly classified. Since it was concluded that the precision of the minority class should be as high as possible, a classification metric based upon the precision should be used for evaluating a classifier's performance. Two classification metrics following this condition are known, the first being the precision itself, and the second the F-measure. The equations for the minority/majority precision, recall, and F-measure are given in equations 7.1 to 7.6. It should, however, be noted that these equations are based upon the minority and majority label in Table 6. If these labels would be turned around, the equations would have to be turned around as well.

Table 6: Recall/precision for minority/majority class

Predicted class / **Actual class**

	Maj	Min
Maj	TP	FN
Min	FP	TN

Recall minority:	R_{min}	$= \frac{TN}{TN + FP}$	[7.1]
Recall majority:	R_{maj}	$= \frac{TP}{TP + FN}$	[7.2]
Precision minority:	P_{min}	$= \frac{TN}{TN + FN}$	[7.3]
Precision majority:	P_{maj}	$= \frac{TP}{TP + FP}$	[7.4]
F-measure minority:	F_{min}	$= \frac{2 * P_{min} * R_{min}}{P_{min} + R_{min}}$	[7.5]
F-measure majority:	F_{maj}	$= \frac{2 * P_{maj} * R_{maj}}{P_{maj} + R_{maj}}$	[7.6]

When comparing these two metrics, it first had to be decided whether the majority class is included or not in determining a classifier's overall performance. Even though the minority class is considered to be the most important class, it was deemed important that the overly present majority class should not be ignored, since overall a classifier is as good as its worst classification performance. A classification metric to indicate the overall performance should therefore focus on both the minority and majority class. To determine which overall performance metric would be best applicable, the pros and cons of using either the F-measure or merely the precision had to be enlisted and weighted, which can be found in Table 7. When using merely the precision as a metric, the overall performance would be measured by calculating the arithmetic mean of the precision of both classes. For determining the overall performance using the F-measure, the macro-average would be used.

Table 7: Pros and cons F-measure and precision

Precision

- Pros
    - A low precision of either class ensures a low overall performance
    - The precision as a classification metric is uninfluenced by any other classification metrics
- Cons
    - A high minority precision can be attained using only a few test cases, making the classifier seem correct, while it is only little informative

F-measure

- Pros
    - A low precision of either class ensures a low overall performance

  - By combining recall and precision, a high overall performance implies that the classifier is very informative
- Cons
  - A high precision of either class could have little influence on the overall performance if the recall is low

The cons as enlisted above will be further explained using Table 8 for the precision and Table 9 for the F-measure. Both tables hold 1000 instances of the majority class and 100 instances of the minority class. In Table 8, all instances of the majority class have been correctly classified, while only 1 instance of the minority class is correct, which results in an average precision of 0.955. Since only one test case is correctly classified as the minority class, the precision of the minority class is 1, while the recall is 0.01. The average precision thus indicates that the classifier is highly informative, while in fact all test cases except for one have been classified as the majority class, which does not add any information. When using the macro-averaged F-measure, as explained in Chapter 6.4.3, the low recall on the minority class negatively influences the overall performance, thereby creating a better perspective on the classifier's overall performance. However, this feature of the macro-averaged F-measure is also its Achilles heel, as a good precision on the minority class, which is deemed to be important for this research, can become overshadowed by a low recall, which will be shown using Table 9. Here, 995 test cases have been correctly classified as the majority class and 30 test cases as the minority class. Even though the precision of the minority class is an acceptable 0.857, the macro-averaged F-measure is 0.704.

Table 8: Example precision con

Predicted class

Actual class		Maj	Min
	Maj	1000	0
	Min	99	1

Table 9: Example F-measure con

Predicted class

Actual class		Maj	Min
	Maj	995	5
	Min	70	30

From the examples it becomes apparent that using merely the precision as an overall classification metric does not hold enough information, and that the macro-averaged F-measure omits to show the influence of the minority precision. Since the precision as an overall classification metric only makes use of the respective individual precisions, the information obtained using this metric cannot be enriched. For the macro-averaged F-measure, however, an alternative is available which could optimize the outcome with respect to the minority precision, namely the $F_{0.5}$-measure. The generic formula for the $F_{0.5}$-measure, which is derived from equation 6.7, is given in equation 7.7 and has been applied to the minority and majority class with respect to Table 6 in equation 7.8 and 7.9. As can be seen from the formulae, the False Positives and False Negatives for respectively the minority and majority class are deemed to be less important with the $F_{0.5}$-measure, thereby increasing the influence of the precision. Using the $F_{0.5}$-measure over the normal F-measure would thus comply better with the preset condition of the minority precision being important, however, the $F_{0.5}$-measure gives a worse representation of the overall performance of a classifier.

$$F_{0.5}\text{-measure} = \frac{(1+0.5^2) * precision * recall}{(0.5^2 * precision) + recall} \qquad [7.7]$$

$$F_{0.5}\text{-measure minority} = \frac{(1+0.5^2) * TN}{(1+0.5^2) * TN + 0.5^2 * FP + FN} \qquad [7.8]$$

$$F_{0.5}\text{-measure majority} = \frac{(1+0.5^2) * TP}{(1+0.5^2) * TP + 0.5^2 * FN + FP} \qquad [7.9]$$

Considering the various downsides in the approaches towards measuring the performance of the results in this research, it was concluded that a single classification metric cannot show both the overall performance as well as the performance on the minority precision. From above observations, it can be concluded that, for this research, the best metric regarding the overall performance is the macro-averaged F-measure (i.e. F_1-measure), following the approach of both Forman (2003) and Yang and Liu (1999). This metric equally weighs both recall and precision of the minority and majority class, thereby clearly showing the overall performance evenly influenced by all variables. To overcome the downside of the F-measure, the macro-averaged $F_{0.5}$-measure will also be used for discussing the performance of a classifier on the minority precision, following the approach of both Ruiz and Srinivasan (1999) and Wang et al. (2014). By combining both metrics, the outcomes of this research will comply with the preset conditions of having a good overall classifier resulting in an as high as possible minority precision.

To determine which classifier holds the best performance, classification results should also be compared to a baseline, since a classification result which does not exceed a baseline holds no additional value (Yang and Liu, 1999). With a skewed dataset, the best uninformed classification results are obtained when classifying all test cases as the majority class, which should then thus be used to calculate a classification metric's baseline. However, since in this research the influence of data alteration techniques on multiple classifiers is assessed, majority voting was considered to be inapplicable. Using majority voting, one would only see how good a classifier performs to when no classifier would be used instead of how much the data alteration techniques influence the performance of the classifier. Therefore, each classifier should be compared to its unimproved counterpart. For both resampling and word weighting this can well be applied, however, for word normalization it was determined to omit test cases where no word normalization was applied for runtime optimization. This required to choose either lemmatization or stemming as the unimproved baseline. As the basis for this decision, the feature list as included in Appendix 12.1 was used, which has randomly been generated using the same train/test set distribution as used for conducting the tests. In this feature list, lemmatization and stemming normalized 36 of the 100 features alike, which was confirmed to be average using three additional randomly generated feature lists. Furthermore, the feature list revealed that using lemmatization, multiple different stems were merged into a single lemma. Since [1] lemmatization more clearly transforms features from their standard form than stemming, thereby having a larger influence on the feature list, and [2] lemmatization and stemming normalize a considerate amount of feature in the same way, it was decided to use stemming as the base form for word normalization. Overall, when evaluating the influence of data alteration techniques on a classifier's performance, it will thus be compared to the results obtained using a training set which has not been sampled, is unweighted, and has been normalized using stemming.

7.2 Research framework

In order to ensure the analyses in this research are performed under equal conditions, a research framework has been created. Analysis is conducted using Weka (Witten and Frank, 2005) combined with R (R Core Team, 2015) using the package RWeka (Hornik et al., 2009). Linguistic analysis is performed by Frog (Van den Bosch et al., 2007), incorporated in the framework using the package Frogr (Van Atteveldt, 2014). This framework consists of two sections: the preprocessing section and the training/testing section.

7.2.1 Preprocessing

The dataset used in this research is an online trade fraud complaints dataset which has been provided by the Dutch National Police and has already been preprocessed by Peters (Peters, 2016a), therefore requiring no additional data preprocessing. Since this research is based upon the prediction of the "withdrawn" label using a complaint's free-text field, these variables are first selected from the dataset, after which a corpus is created where each document represented a single complaint. The words in each document are transformed to their base form using either a Dutch adapted stemmer (Porter, 1980) or lemmatizer (Van den Bosch et al., 2007). In a subsequent cleaning phase, all punctuation is removed and Dutch stop words are removed according to a predefined list[14]. Next, the corpus is split using stratified 10-fold cross validation, so that the ratio of minority to majority classes in each fold is maintained and each complaint is used once for testing and nine times for training.

7.2.2 Training/testing

Each of the 10 folds resulting from the preprocessing section is used for training and testing a classifier following the same procedure as illustrated in Figure 11. First, 90% of the fold

[14] http://snowball.tartarus.org/algorithms/dutch/stop.txt

assigned for training the classifier is transformed into a term-document matrix containing 1:3-grams. All terms present in the term-document matrix are evaluated for their presence in the documents (i.e. complaints) and the term-document matrix is reduced to the 100 most occurring terms. Next, the term-document matrix is resampled using either random undersampling or SMOTE (Chawla et al., 2002). With random undersampling, the minority class entries are kept constant and the majority class entries are randomly downscaled according to the training sample distribution. For SMOTE, the minority class entries are upscaled to match the amount of majority class entries and the majority class entries are randomly up- or downscaled according to the training sample distribution. The classifier is then build on the resampled term-document matrix using Weka's default classifier implementations (Frank et al., 2010). To avoid interference with the aspects under investigation in this research, we have opted to use the basic parameter settings provided by Weka.

After the classifier has been built it is tested using the assigned 10% of the fold. A term-document matrix is first created on all documents in the testing corpus, after which it is reduced to the same 100 terms used for training the classifier. If a term used for training the classifier does not occur in the term-document matrix of the test set, a value of 0 is assigned to it for each document indicating its absence. When the term-document matrix has been constructed it is entered as input to the classifier and the predicted class labels are evaluated with respect to the actual class labels.

Finally, the evaluation results of each individual fold are combined and averaged resulting in an overall evaluation for the classifier under the predefined settings.

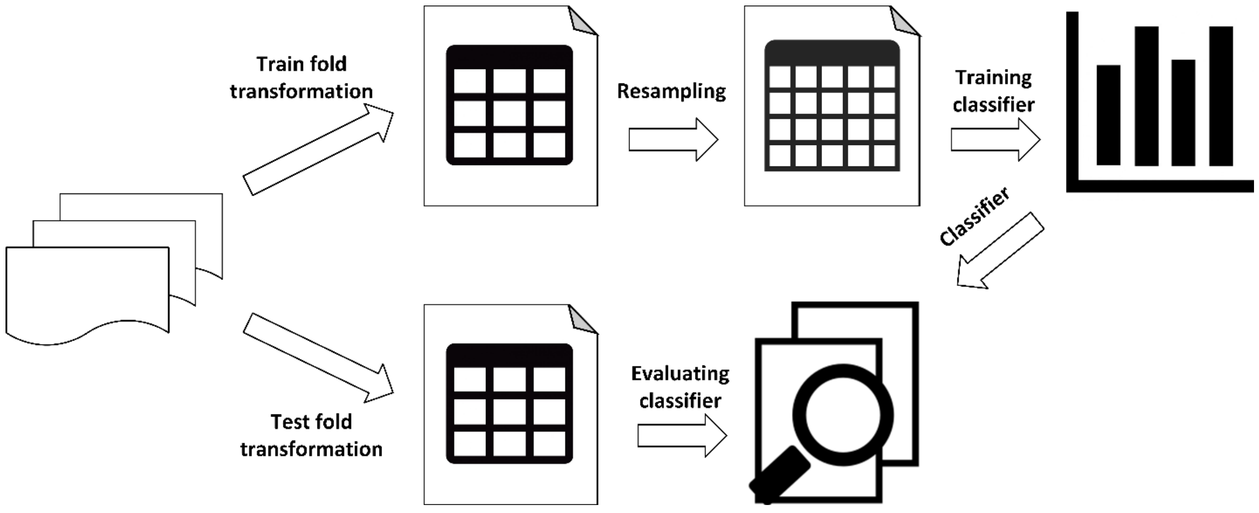


Figure 11: Training/testing section in research framework

7.3 Multinomial naive Bayes

Table 10 holds an overview of the macro-averaged F_1-measures obtained using the default implementation of WEKA's NaiveBayesMultinomial classifier (McCallum and Nigam, 1998). The macro-averaged $F_{0.5}$-measures are included in between brackets to more clearly distinct the influence of the precision on the F-measure. For each normalization/weighting technique combination, the best results are displayed in bold and the overall best result for a combination is underlined.

In Figure 12 the F-measures of the minority and majority class, using undersampling and lemmatization, are illustrated, while Figure 13 depicts the recall and precision. A complete overview of the illustrations can be found in Appendix 12.2.

Table 10: F-measures using multinomial naive Bayes

Resampling technique	**Normalization technique**	**Training sample distribution**	**Weighting technique**		
			No Weighting	Binary	TF-IDF
No Sampling	Lemmatization	16.7:83.3	<u>**0.583**</u> (0.588)	0.557 (0.573)	0.454 (0.431)
	Stemming	16.7:83.3	<u>**0.572**</u> (0.579)	0.553 (0.569)	0.454 (0.431)
Undersampling	Lemmatization	30:70	**0.589** (0.584)	<u>**0.590**</u> (0.588)	0.454 (0.431)
		40:60	0.576 (0.573)	0.585 (0.579)	0.455 (0.433)
		50:50	0.545 (0.553)	0.550 (0.556)	**0.539** (0.548)
		60:40	0.509 (0.531)	0.503 (0.527)	0.146 (0.105)
		70:30	0.462 (0.450)	0.440 (0.485)	0.144 (0.101)
	Stemming	30:70	<u>**0.588**</u> (0.579)	**0.584** (0.583)	0.454 (0.431)
		40:60	0.569 (0.567)	0.576 (0.572)	0.455 (0.432)
		50:50	0.537 (0.545)	0.544 (0.550)	**0.534** (0.544)
		60:40	0.496 (0.521)	0.493 (0.519)	0.146 (0.106)
		70:30	0.453 (0.492)	0.432 (0.478)	0.144 (0.101)
Oversampling	Lemmatization	30:70	<u>**0.584**</u> (0.580)	**0.583** (0.581)	0.454 (0.431)

		40:60	0.567 (0.566)	0.573 (0.569)	0.455 (0.433)
		50:50	0.541 (0.549)	0.550 (0.554)	**0.548** (0.554)
		60:40	0.509 (0.530)	0.509 (0.529)	0.147 (0.108)
		70:30	0.465 (0.501)	0.453 (0.493)	0.143 (0.101)
	Stemming	30:70	**0.578** (0.574)	**<u>0.579</u>** (0.578)	0.454 (0.431)
		40:60	0.558 (0.557)	0.566 (0.563)	0.455 (0.432)
		50:50	0.533 (0.542)	0.538 (0.545)	**0.542** (0.549)
		60:40	0.497 (0.520)	0.496 (0.520)	0.147 (0.109)
		70:30	0.455 (0.492)	0.442 (0.484)	0.144 (0.101)

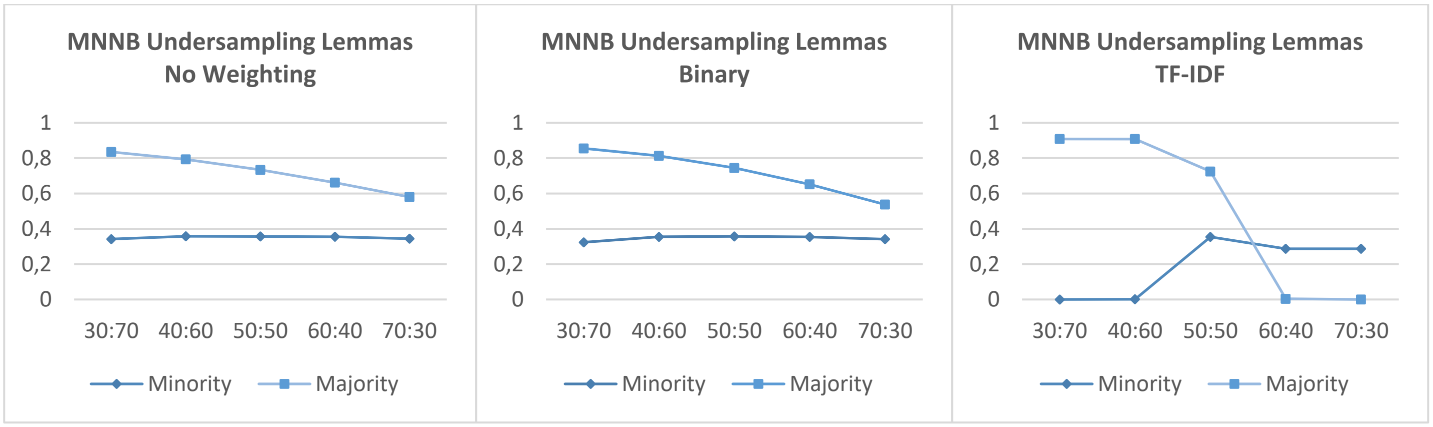


Figure 12: F-measures using multinomial naive Bayes;
X-axis: training sample distribution, Y-axis: F-measure

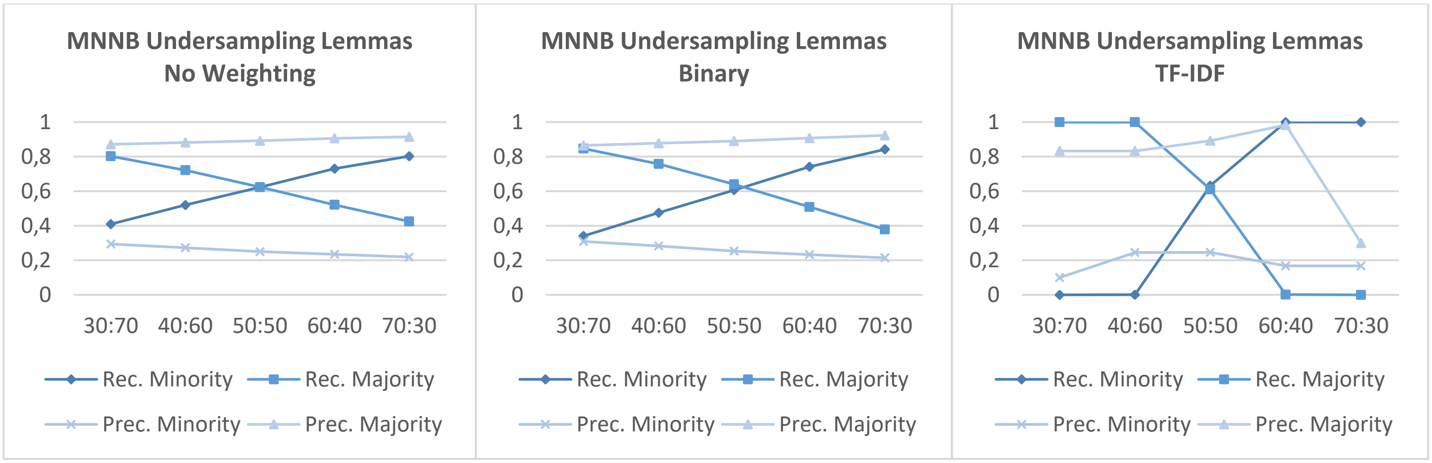


Figure 13: Recall and precision using multinomial naive Bayes;
X-axis: training sample distribution, Y-axis: recall/precision

7.3.1 General results

Overall, the best classification result is achieved using an undersampled 30:70 training distribution with lemmatization and binary weighting. Here, the F-measure of the majority class is reduced by 1.6% with respect to the baseline (87.2%) and the F-measure of the minority class is improved with 5.2%. In general, a training sample distribution which resembles the original distribution tends to hold the best classification performance.

7.3.2 Resampling technique

For evaluating the overall performance of resampling techniques, not only do the actual techniques have to be compared, but also the training sample distributions. In general, undersampling tends to outperform oversampling for training sample distributions in which the majority class is dominant up to 50:50. When the minority class becomes the dominant class in the training sample distribution, this changes to oversampling outperforming undersampling. However, the difference between the over- and undersampling is of minor influence on the performance of a multinomial naive Bayes classifier. A greater difference can be noticed between the training sample distributions, where a distribution with a dominant majority class shows a better performance. When evaluating the recall and precision, it can be noticed that, as more minority cases are included in the training sample, the recall of the minority class improves while the recall of the majority class decreases for both binary and no weighting. Meanwhile, the precision of the minority and majority class are not influenced, resulting in a decreased F-measure for the majority class and an almost even F-measure for the minority class.

7.3.3 Word normalization technique

When classifying the test set using multinomial naive Bayes, it can be noticed that lemmatization consistently outperforms stemming with respect to the F-measure for all resampling and word weighting techniques. Inducing this outperformance, both recall and precision are slightly improved when using lemmatization.

7.3.4 Word weighting technique

In general, both no weighting and binary weighting tend to have a comparable classification performance for both under- and oversampling when using a 30:70 training distribution. For distributions 40:60 and 50:50, binary weighting slightly outperforms no weighting, while for distributions 60:40 and 70:30 no weighting slightly outperforms binary weighting. When applying no sampling to the training set, it can be observed that using no weighting on the term-document matrix benefits the performance of a multinomial naive Bayes classifier. For TF-IDF weighting, an interesting finding can be observed. When applying a 30:70 and 40:60 training sample distribution for both under- and oversampling, all instances are classified as the majority class due to the profound effect of the majority a priori probability on the class prediction, resulting in a recall of 1 for the majority class and a recall of 0 for the minority class. For a 60:40 and 70:30 distribution opposed results can be observed. Arising from these observations is an F-measure of 0 for the class which has not been predicted. With a 50:50 training sample distribution, this phenomenon does not occur, resulting in a higher average F-measure. Using a TF-IDF weighting, however, does not pose the best classification results.

7.4 Logistic regression

Table 11 holds an overview of the average F-measures obtained using the default implementation of WEKA's Logistic classifier (Le Cessie and van Houwelingen, 1992). In Figure 14 the F-measures of the minority and majority class, using undersampling and lemmatization, are illustrated, while Figure 15 depicts the recall and precision. A complete overview of the illustrations can be found in Appendix 12.3.

Table 11: F-measures using logistic regression

Resampling technique	**Normalization technique**	**Training sample distribution**	**Weighting technique**		
			No Weighting	Binary	TF-IDF
No Sampling	Lemmatization	16.7:83.3	**<u>0.468</u>** (0.462)	0.467 (0.460)	0.456 (0.436)
	Stemming	16.7:83.3	**<u>0.466</u>** (0.458)	0.464 (0.455)	0.456 (0.435)
Undersampling	Lemmatization	30:70	0.545 (0.565)	0.562 (0.581)	0.520 (0.538)
		40:60	**<u>0.594</u>** (0.590)	**0.593** (0.588)	**0.593** (0.589)
		50:50	0.542 (0.552)	0.544 (0.553)	0.538 (0.550)
		60:40	0.430	0.451	0.417

			(0.477)	(0.493)	(0.467)
		70:30	0.311 (0.370)	0.341 (0.401)	0.295 (0.353)
	Stemming	30:70	0.532 (0.550)	0.544 (0.561)	0.509 (0.525)
		40:60	**0.583** (0.581)	**<u>0.583</u>** (0.579)	**0.581** (0.577)
		50:50	0.532 (0.544)	0.534 (0.546)	0.529 (0.542)
		60:40	0.422 (0.470)	0.443 (0.486)	0.412 (0.463)
		70:30	0.307 (0.366)	0.329 (0.390)	0.292 (0.349)
Oversampling	Lemmatization	30:70	0.554 (0.572)	0.571 (0.578)	0.565 (0.576)
		40:60	**0.582** (0.577)	**0.577** (0.573)	**<u>0.588</u>** (0.583)
		50:50	0.528 (0.541)	0.541 (0.549)	0.550 (0.556)
		60:40	0.452 (0.492)	0.484 (0.514)	0.475 (0.508)
		70:30	0.367 (0.425)	0.411 (0.462)	0.383 (0.439)
	Stemming	30:70	0.538 (0.553)	0.559 (0.566)	0.553 (0.562)
		40:60	**0.569** (0.566)	**0.565** (0.562)	**<u>0.579</u>** (0.575)
		50:50	0.521 (0.534)	0.533 (0.542)	0.542 (0.549)
		60:40	0.447 (0.487)	0.477 (0.508)	0.476 (0.508)
		70:30	0.366 (0.423)	0.407 (0.458)	0.383 (0.439)

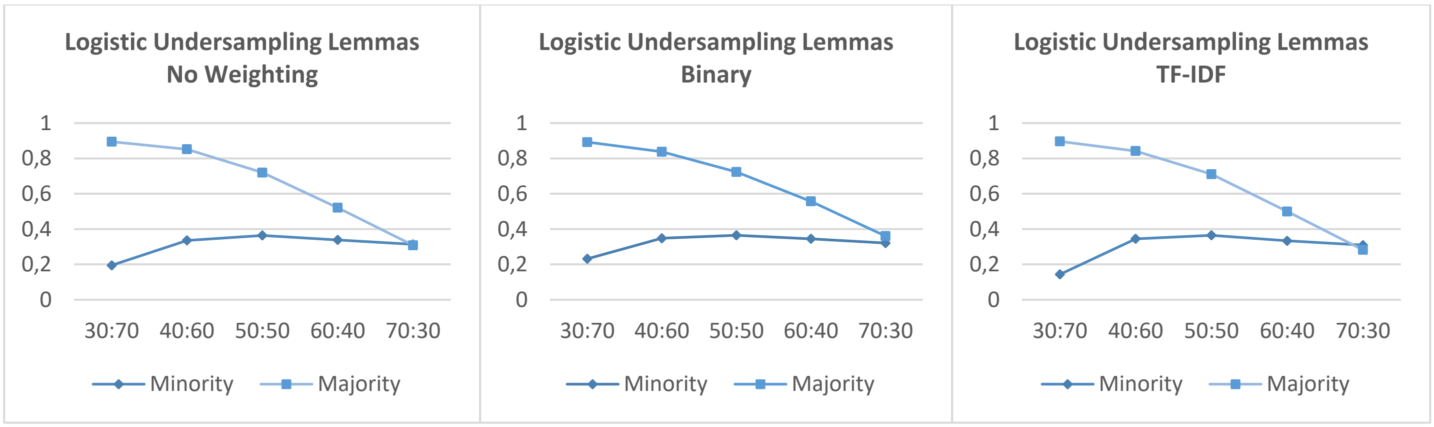


Figure 14: F-measures using logistic regression;
X-axis: training sample distribution, Y-axis: F-measure

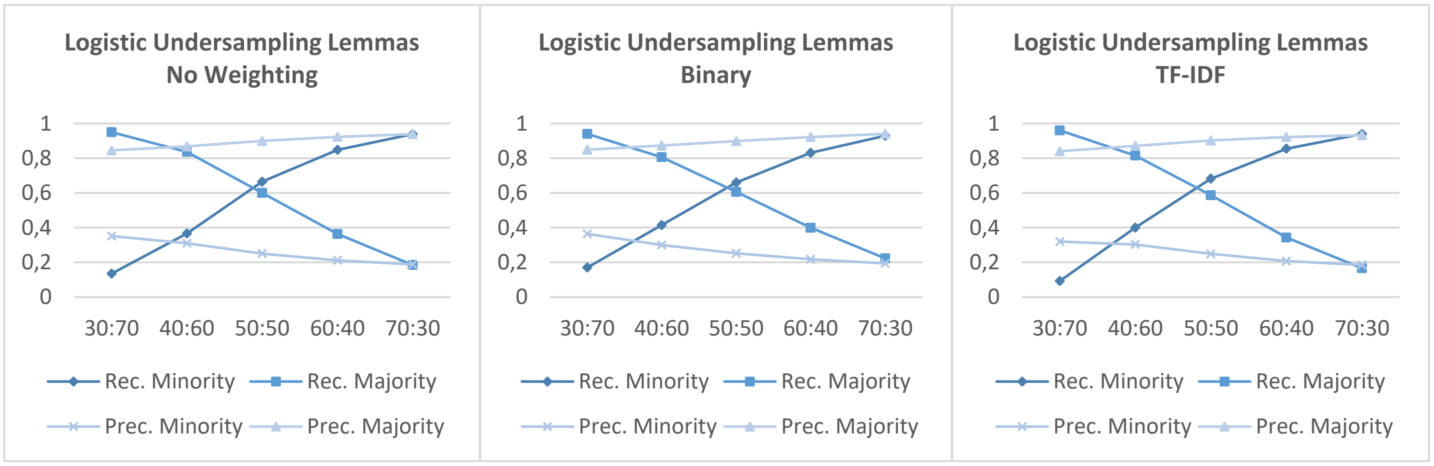


Figure 15: Recall and precision using logistic regression;
X-axis: training sample distribution, Y-axis: recall/precision

7.4.1 General results

For logistic regression, the best classification result is achieved using an undersampled 40:60 distribution with lemmatization and no weighting. Here, the F-measure of the majority class is reduced by 5.6% with respect to the baseline (90.7%) and the F-measure of the minority class is improved with 33.4%. Overall, resampling a dataset before training a logistic regression classifier can improve its F-measure up to 12.8%, with a 40:60 training sample distribution resulting in the best performance, independent of the weighting technique applied.

7.4.2 Resampling technique

Both under- and oversampling consistently outperform one another on specific training sample distributions, independent of the weighting technique applied. For a 40:60 or 50:50 distribution it is best to use undersampling, where for the other distributions oversampling has the best performance. The classification results furthermore show that the average F-measure of a logistic regression classifier improves as the minority cases are scaled up to 40%. When evaluating the recall and precision, it can be observed that adding more minority cases greatly benefits the recall of the minority class, while removing majority cases only slightly decreases the recall of the majority class. Starting at a 50:50 distribution, this effect is reversed and the average F-measure decreases. For undersampling this effect is stronger noticeable than for oversampling.

7.4.3 Word normalization technique

With respect to lemmatization and stemming, similar results are observed as with multinomial naive Bayes.

7.4.4 Word weighting technique

In general, applying binary weighting tends to outperform both TF-IDF and no weighting for all resampling and word normalization techniques. When comparing TF-IDF and no weighting, it is observed that applying no weighting benefits the F-measure of a logistic regression classifier when undersampling and applying TF-IDF weighting when oversampling. Applying TF-IDF weighting positively influences any small values regarding the recall and precision of a logistic regression classifier. Furthermore, while reducing both the recall of the minority class and the precision of the majority class in comparison to applying no weighting, the recall of the majority class and precision of the minority class are increased. Since the F-measure is a harmonic mean, the influence of the increased smaller values outweighs the influence of the decreased larger values, thus resulting in a better overall performance.

7.5 Decision tree

Table 12 holds an overview of the average F-measures obtained using the default implementation of WEKA's J48 classifier (Quinlan, 1993). In Figure 16 the F-measures of the minority and majority class, using undersampling and lemmatization, are illustrated, while Figure 17 depicts the recall and precision. A complete overview of the illustrations can be found in Appendix 12.4.

Table 12: F-measures using decision tree

Resampling technique	**Normalization technique**	**Training sample distribution**	**Weighting technique**		
			No Weighting	Binary	TF-IDF
No Sampling	Lemmatization	16.7:83.3	**<u>0.536</u>** (0.551)	0.535 (0.551)	0.511 (0.526)
	Stemming	16.7:83.3	**<u>0.527</u>** (0.540)	0.523 (0.537)	0.500 (0.510)
Undersampling	Lemmatization	30:70	**0.552** (0.549)	**0.550** (0.548)	**<u>0.560</u>** (0.558)

		40:60	0.533 (0.536)	0.535 (0.537)	0.542 (0.544)
		50:50	0.497 (0.513)	0.496 (0.512)	0.504 (0.522)
		60:40	0.456 (0.489)	0.454 (0.487)	0.437 (0.477)
		70:30	0.394 (0.444)	0.395 (0.446)	0.382 (0.436)
	Stemming	30:70	**0.544** (0.542)	**0.545** (0.543)	**0.549** (0.548)
		40:60	0.532 (0.535)	0.530 (0.533)	0.534 (0.538)
		50:50	0.494 (0.522)	0.496 (0.512)	0.491 (0.511)
		60:40	0.445 (0.477)	0.447 (0.482)	0.423 (0.466)
		70:30	0.388 (0.436)	0.386 (0.438)	0.378 (0.432)
Oversampling	Lemmatization	30:70	0.537 (0.541)	0.537 (0.542)	0.540 (0.538)
		40:60	0.542 (0.543)	**0.546** (0.547)	**0.536** (0.535)
		50:50	**0.543** (0.542)	0.546 (0.544)	0.528 (0.530)
		60:40	0.541 (0.540)	0.539 (0.538)	0.513 (0.520)
		70:30	0.525 (0.529)	0.523 (0.527)	0.492 (0.509)
	Stemming	30:70	0.529 (0.534)	0.532 (0.537)	0.528 (0.527)
		40:60	0.534 (0.535)	0.538 (0.539)	**0.532** (0.532)
		50:50	**0.539** (0.538)	**0.539** (0.538)	0.519 (0.522)
		60:40	0.533 (0.532)	0.534 (0.533)	0.511 (0.519)
		70:30	0.518 (0.522)	0.519 (0.523)	0.488 (0.506)

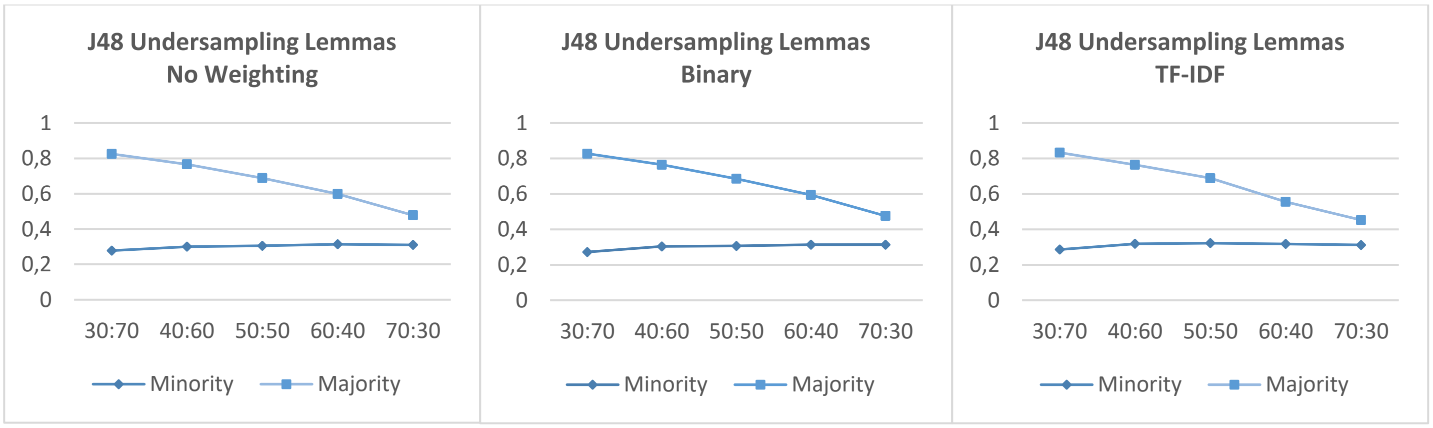


Figure 16: F-measures using decision tree;
X-axis: training sample distribution, Y-axis: F-measure

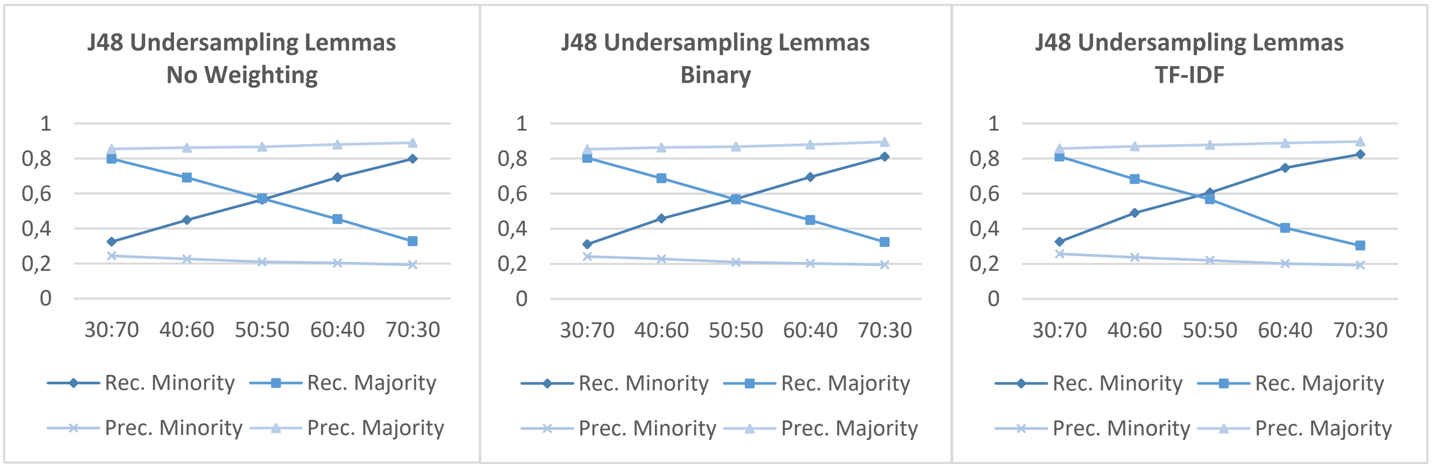


Figure 17: Recall and precision using decision tree;
X-axis: training sample distribution, Y-axis: recall/precision

7.5.1 General results

Using a decision tree classifier, the best classification result is achieved for an undersampled 30:70 training sample distribution with lemmatization and TF-IDF weighting. In comparison to the baseline (88.7%), the F-measure of the majority class is reduced by 5.4%, but the F-measure of the minority class is improved with 12%. Overall, resampling and weighting a dataset before training a decision tree classifier consistently improves its F-measure up to 3.3%.

7.5.2 Resampling technique

When undersampling a dataset, the best classification results are consistently achieved using a 30:70 training sample distribution, closely resembling the distribution of the original dataset. By diminishing the amount of majority cases in the training set, the F-measure of the minority class is barely influenced, while the F-measure of the majority class plummets. For oversampling a different pattern can be observed where the F-measure of the majority class only slightly decreases by changing the distribution. Since the F-measure of the minority class increases at nearly the same rate, the average F-measure appears to be fixed. When comparing the recall of both under- and oversampling, it can be found that for oversampling the recall of the minority and majority class starts at a respectively lower and higher rate than for undersampling and shows a smaller increase and decrease. For oversampling less test cases are thus classified as the minority class when a greater training sample distribution is used resulting in a more constant F-measure.

7.5.3 Word normalization technique

With respect to lemmatization and stemming, similar results are observed as with multinomial naive Bayes.

7.5.4 Word weighting technique

In general, both binary and no weighting tend to equally influence the F-measure of a decision tree classifier. It is therefore interesting to look at the influence of TF-IDF weighting in comparison to binary and no weighting for both under- and oversampling. TF-IDF weighting consistently shows the best performance when undersampling with a 30:70 or 40:60 training sample distribution, but underperforms for a 50:50 distribution and above. Underlying this pattern is a higher recall and precision for the minority class in combination with an even recall and precision for the majority class. As the training sample distribution increases, the recall of the majority class decreases in comparison to binary and no weighting, resulting in a lower F-measure. For oversampling a similar pattern emerges, however, instead of outperforming binary and no weighting, TF-IDF shows a similar performance with a 30:70 or 40:60 training sample distribution, which is explained by a lower recall on the majority class, resulting in a lower average F-measure.

7.6 Multivariate naive Bayes

Table 13 holds an overview of the average F-measures obtained using the default implementation of WEKA's NaiveBayes classifier (John and Langley, 1995). In Figure 18 the F-measures of the minority and majority class, using undersampling and lemmatization, are illustrated, while Figure 19 depicts the recall and precision. A complete overview of the illustrations can be found in Appendix 12.5.

Table 13: F-measures using multivariate naive Bayes

Resampling technique	**Normalization technique**	**Training sample distribution**	**Weighting technique**		
			No Weighting	Binary	TF-IDF
No Sampling	Lemmatization	16.7:83.3	0.573 (0.570)	**<u>0.586</u>** (0.583)	0.508 (0.525)
	Stemming	16.7:83.3	0.560	**<u>0.581</u>**	0.452

			(0.561)	(0.579)	(0.488)
Undersampling	Lemmatization	30:70	**0.552** (0.556)	**0.573** (0.571)	**0.486** (0.512)
		40:60	0.526 (0.540)	0.555 (0.559)	0.484 (0.511)
		50:50	0.508 (0.529)	0.534 (0.546)	0.471 (0.502)
		60:40	0.484 (0.513)	0.513 (0.533)	0.473 (0.503)
		70:30	0.453 (0.492)	0.485 (0.515)	0.471 (0.502)
	Stemming	30:70	**0.527** (0.541)	**0.566** (0.564)	**0.434** (0.476)
		40:60	0.508 (0.527)	0.552 (0.555)	0.421 (0.466)
		50:50	0.486 (0.513)	0.531 (0.542)	0.418 (0.464)
		60:40	0.464 (0.498)	0.510 (0.529)	0.408 (0.457)
		70:30	0.446 (0.486)	0.484 (0.513)	0.340 (0.450)
Oversampling	Lemmatization	30:70	**0.453** (0.482)	**0.509** (0.518)	**0.431** (0.471)
		40:60	0.450 (0.480)	0.507 (0.517)	0.428 (0.468)
		50:50	0.448 (0.479)	0.505 (0.516)	0.425 (0.467)
		60:40	0.446 (0.477)	0.503 (0.515)	0.425 (0.466)
		70:30	0.443 (0.476)	0.501 (0.514)	0.417 (0.461)
	Stemming	30:70	**0.453** (0.482)	**0.504** (0.515)	**0.423** (0.465)
		40:60	0.451 (0.481)	0.503 (0.515)	0.420 (0.463)
		50:50	0.448 (0.479)	0.501 (0.513)	0.417 (0.461)
		60:40	0.447 (0.478)	0.500 (0.513)	0.413 (0.458)
		70:30	0.444 (0.477)	0.498 (0.512)	0.410 (0.455)

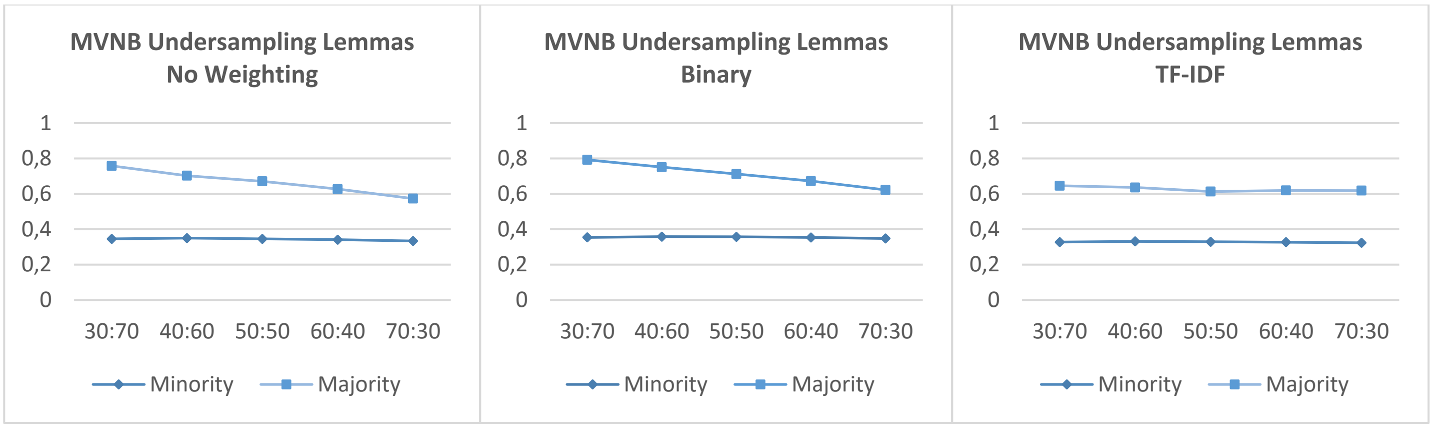


Figure 18: F-measures using multivariate naive Bayes;
X-axis: training sample distribution, Y-axis: F-measure

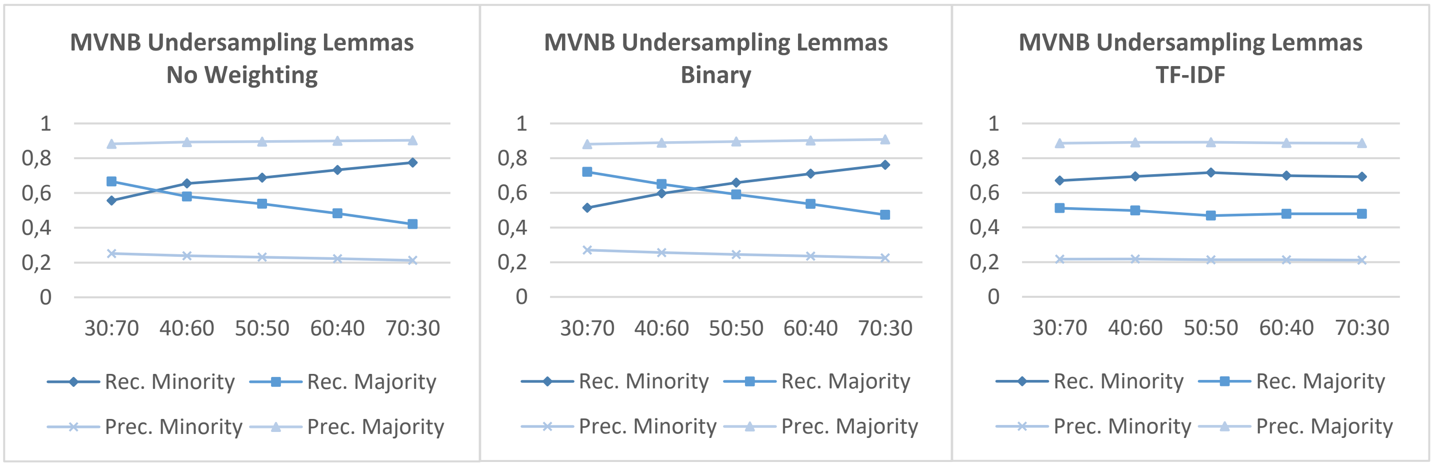


Figure 19: Recall and precision using multivariate naive Bayes;
X-axis: training sample distribution, Y-axis: recall/precision

7.6.1 General results

Overall, the best classification result is achieved using no sampling and applying binary weighting. Since resampling does not improve the base classifier, the individual effects of the data alteration techniques will be assessed with respect to the unsampled multivariate naive Bayes classifier.

7.6.2 Resampling technique

When training a multivariate naive Bayes classifier, resampling the training set consistently reduces the F-measure of the majority class with respect to no sampling, which results from a reduced recall on the majority class. Even though the recall of the minority class is clearly improved, the precision remains the same thus resulting in an unimproved F-measure for the minority class. Resampling the training set causes more majority instances to be classified as minority instances and thereby lessening the majority F-measure without enhancing the minority F-measure.

7.6.3 Word normalization technique

With respect to lemmatization and stemming, similar results are observed as with multinomial naive Bayes.

7.6.4 Word weighting technique

Even though the average F-measure of binary and no weighting tends to be equal, the individual F-measures of the minority and majority class differ substantially, which can also be observed with the recall and precision. Where applying no weighting results in a higher minority recall and lower majority recall, applying binary weighting results in a lower minority recall and higher majority recall. Since the precision of the majority class is higher than that of the minority class, the effect of a shift in the majority class recall outweighs the effect of a shift in the minority class recall on the majority F-measure. For TF-IDF, the decrease in majority class recall is even larger than with no weighting, causing a greater effect on the F-measure.

7.7 Association rule

Table 14 holds an overview of the average F-measures obtained using the default implementation of WEKA's JRip classifier (Cohen, 1995). In Figure 20 the F-measures of the minority and majority class, using undersampling and lemmatization, are illustrated, while Figure 21 depicts the recall and precision. A complete overview of the illustrations can be found in Appendix 12.6.

Table 14: F-measures using RIPPER

Resampling technique	**Normalization technique**	**Training sample distribution**	**Weighting technique**		
			No Weighting	Binary	TF-IDF
No Sampling	Lemmatization	16.7:83.3	0.457 (0.437)	0.459 (0.442)	**<u>0.459</u>** (0.443)
	Stemming	16.7:83.3	0.456 (0.435)	**<u>0.458</u>** (0.441)	0.456 (0.436)
Undersampling	Lemmatization	30:70	0.566 (0.576)	0.565 (0.575)	0.559 (0.571)
		40:60	**<u>0.585</u>** (0.581)	**0.581** (0.577)	**0.576** (0.573)
		50:50	0.539 (0.548)	0.540 (0.548)	0.531 (0.541)
		60:40	0.446 (0.487)	0.443 (0.486)	0.444 (0.486)
		70:30	0.320 (0.381)	0.332 (0.399)	0.314 (0.374)
	Stemming	30:70	0.546	0.547	0.543

			(0.559)	(0.558)	(0.559)
		40:60	**0.574** (0.570)	**0.571** (0.568)	**0.578** (0.577)
		50:50	0.532 (0.541)	0.530 (0.539)	0.527 (0.537)
		60:40	0.441 (0.484)	0.439 (0.483)	0.432 (0.477)
		70:30	0.291 (0.351)	0.299 (0.358)	0.281 (0.336)
Oversampling	Lemmatization	30:70	0.454 (0.431)	0.454 (0.431)	0.487 (0.496)
		40:60	0.454 (0.431)	0.454 (0.431)	0.502 (0.516)
		50:50	0.454 (0.431)	0.466 (0.457)	0.521 (0.536)
		60:40	**0.546** (0.544)	**0.547** (0.547)	**0.527** (0.533)
		70:30	0.526 (0.533)	0.521 (0.531)	0.464 (0.495)
	Stemming	30:70	0.454 (0.431)	0.454 (0.431)	0.481 (0.486)
		40:60	0.454 (0.431)	0.454 (0.431)	0.497 (0.508)
		50:50	0.454 (0.431)	0.457 (0.431)	**0.523** (0.534)
		60:40	**0.538** (0.538)	**0.538** (0.539)	0.523 (0.530)
		70:30	0.514 (0.523)	0.501 (0.520)	0.447 (0.483)

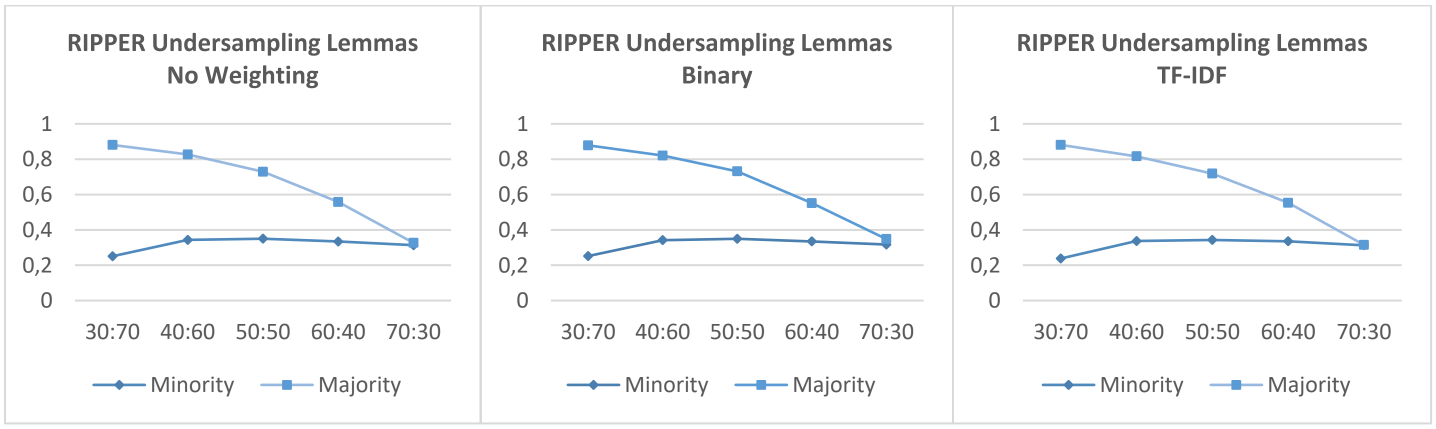


Figure 20: F-measures using RIPPER;
X-axis: training sample distribution, Y-axis: F-measure

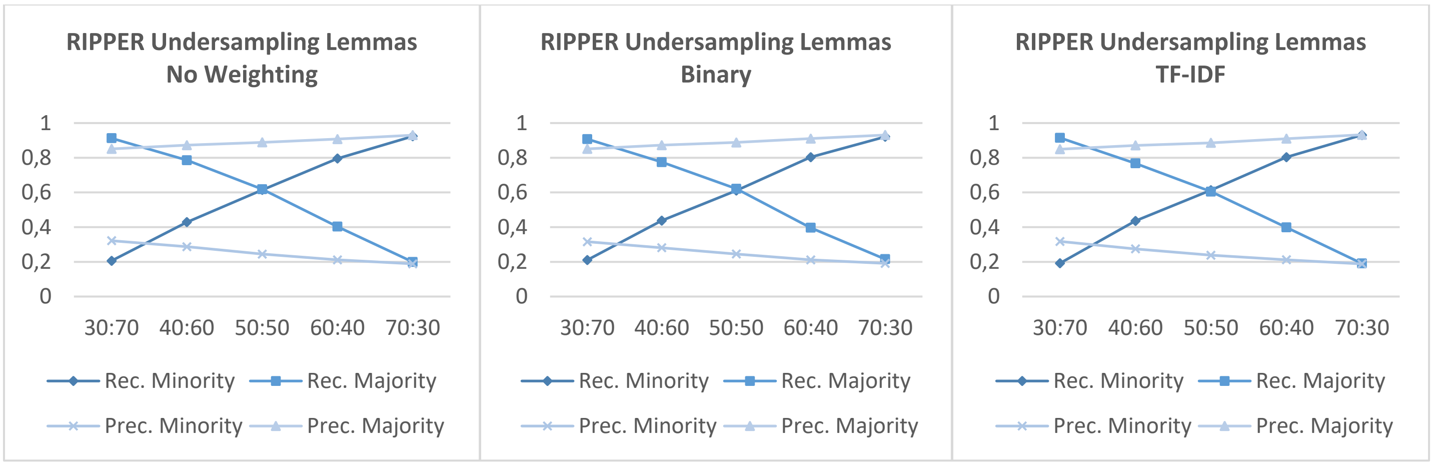


Figure 21: Recall and precision using RIPPER;
X-axis: training sample distribution, Y-axis: recall/precision

7.7.1 General results

Using an association rule classifier, the best classification result is achieved for an undersampled 40:60 training sample distribution with lemmatization and no weighting. Here, the F-measure of the minority class is improved with 34.3% with respect to the baseline (0%) and the F-measure of the majority class is decreased with 8.1%. By both resampling and weighting a training set, the average F-measure of an association rule classifier can be improved with up to 12.9%.

7.7.2 Resampling technique

Under- and oversampling both show a different pattern on the training sample distributions. With undersampling, the average F-measure first improves when increasing the training sample distribution to 40:60 after which the average F-measure declines. This initial rise in F-measure can be explained by an initially greater improvement of the minority class recall and majority class precision in comparison to the decline in majority class recall and minority class precision. Starting from a 50:50 training sample distribution these patterns reverse and the F-measure of the majority class decays due to a lower recall. When oversampling a training set, an association rule classifier predicts most test cases as the majority class up to a 50:50 training sample distribution due to the excessive presence of the majority cases in the training set. By increasing the proportion of minority cases, the test entities will be more and more classified as the minority class, resulting in a higher minority class recall and a lower majority class recall. Overall, undersampling performs best for training sample distributions up to 50:50 and oversampling for the distributions with a greater proportion of minority cases.

7.7.3 Word normalization technique

With respect to lemmatization and stemming, similar results are observed as with multinomial naive Bayes.

7.7.4 Word weighting technique

For undersampling, no general patterns regarding the applied word weighting technique can be observed. All word weighting techniques tend to have a similar performance on all training sample distributions, with TF-IDF having the lowest F-measure. When oversampling, the influence of TF-IDF weighting is compared to both binary and no weighting, since they result in similar F-measures. TF-IDF weighting consistently shows the best performance when oversampling a training set up to a 50:50 training sample distribution, but underperforms for a 60:40 or 70:30 distribution. Where an association rule classifier using binary or no weighting is highly influenced by the excessive amount of majority cases in training sample distributions up to 50:50, a classifier using TF-IDF weighting is still able to predict test cases as the minority class. However, as the distribution shifts upwards, the recall of the TF-IDF weighted classifier decays, resulting in a lower average F-measure.

7.8 Neural network

Table 15 holds an overview of the average F-measures obtained using the default implementation of WEKA's MultilayerPerceptron classifier with the maximum number of epochs capped at 100 due to runtime issues. In Figure 22 the F-measures of the minority and majority class, using undersampling and lemmatization, are illustrated, while Figure 23 depicts the recall and precision. A complete overview of the illustrations can be found in Appendix 12.7.

Table 15: F-measures using neural network

Resampling technique	**Normalization technique**	**Training sample distribution**	**Weighting technique**		
			No Weighting	Binary	TF-IDF
No Sampling	Lemmatization	16.7:83.3	0.454 (0.431)	**<u>0.509</u>** (0.526)	0.454 (0.431)
	Stemming	16.7:83.3	0.454 (0.431)	**<u>0.492</u>** (0.504)	0.454 (0.431)
Undersampling	Lemmatization	30:70	**0.530** (0.558)	**<u>0.564</u>** (0.564)	0.457 (0.436)
		40:60	0.526 (0.557)	0.538 (0.544)	**0.562** (0.570)
		50:50	0.421 (0.480)	0.513 (0.530)	0.506 (0.534)
		60:40	0.509 (0.570)	0.481 (0.511)	0.527 (0.546)
		70:30	0.269 (0.324)	0.407 (0.458)	0.214 (0.242)
	Stemming	30:70	0.517 (0.544)	**0.551** (0.554)	0.455 (0.433)

		40:60	**0.523** (0.567)	0.530 (0.541)	**0.558** (0.560)
		50:50	0.503 (0.545)	0.508 (0.529)	0.465 (0.503)
		60:40	0.505 (0.546)	0.508 (0.529)	0.520 (0.543)
		70:30	0.259 (0.350)	0.377 (0.433)	0.205 (0.228)
Oversampling	Lemmatization	30:70	0.435 (0.544)	**0.552** (0.551)	0.175 (0.247)
		40:60	**0.505** (0.588)	0.551 (0.579)	**0.456** (0.436)
		50:50	0.463 (0.487)	0.544 (0.544)	0.175 (0.239)
		60:40	0.385 (0.459)	0.524 (0.531)	0.216 (0.358)
		70:30	0.366 (0.446)	0.500 (0.517)	0.151 (0.119)
	Stemming	30:70	0.352 (0.461)	0.541 (0.543)	0.176 (0.298)
		40:60	**0.468** (0.537)	**0.545** (0.545)	**0.455** (0.432)
		50:50	0.334 (0.445)	0.531 (0.534)	0.176 (0.265)
		60:40	0.343 (0.380)	0.528 (0.533)	0.179 (0.262)
		70:30	0.251 (0.230)	0.481 (0.507)	0.159 (0.140)

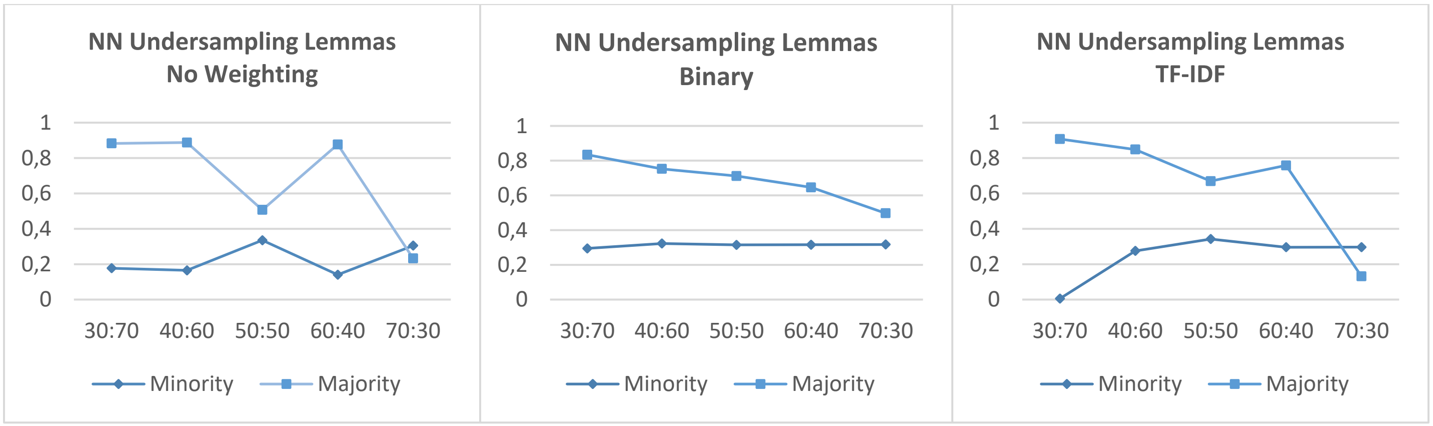


Figure 22: F-measures using neural network;
X-axis: training sample distribution, Y-axis: F-measure

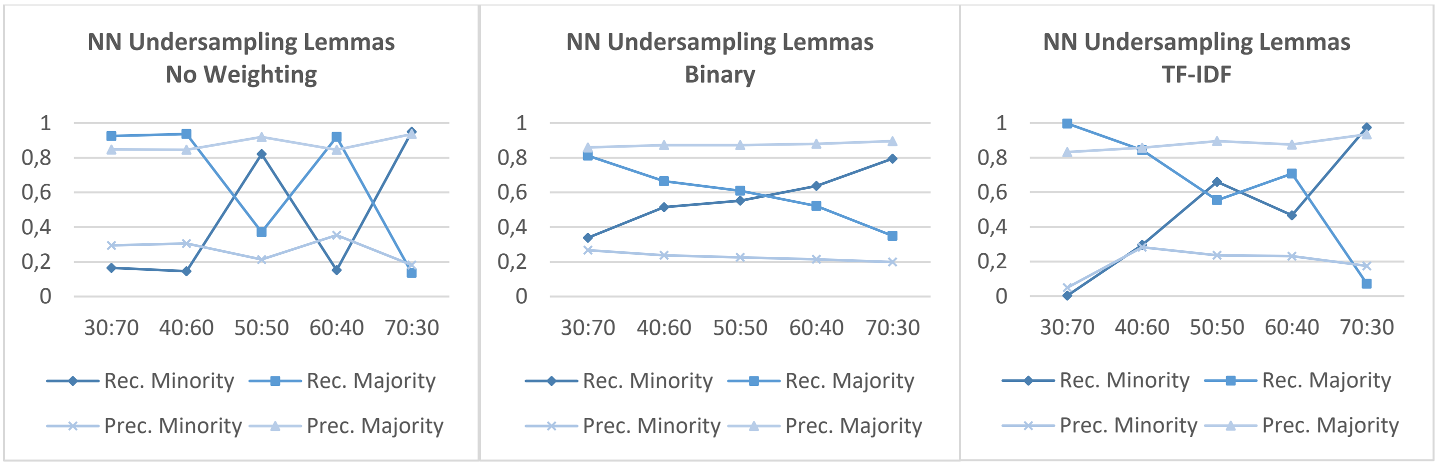


Figure 23: Recall and precision using neural network;
X-axis: training sample distribution, Y-axis: recall/precision

7.8.1 General results

Overall, the best classification result is achieved using an undersampled 30:70 training distribution with lemmatization and binary weighting. Here, the F-measure of the majority class is reduced by 7.5% with respect to the baseline (90.9%) and the F-measure of the minority class is improved with 29.5%. Since no general patterns regarding classifier performance can be observed due to highly alternating results, it has been decided to combine all findings into a single chapter.

A first pattern which can be observed between under- and oversampling is the difference between all graphs in Appendix 10.7. Applying either under- or oversampling results in different local minima, which can clearly be seen by the highly fluctuating graphs. An underlying reason for this could be that undersampling is performed at random with no test cases being duplicated, while with oversampling, minority test cases are duplicated using SMOTE. When a single test case, which has often been duplicated, has a great influence on the algorithm, the neural network classifier could be forced into a local minimum for either minority or majority cases.

When undersampling, the results obtained for the application of no weighting and TF-IDF weighting tend to follow a similar pattern. For a 30:70 or 40:60 training sample distribution, a high recall for the majority class and low recall for the minority class are obtained. A local minimum for a 50:50 distribution causes more instances to be classified as the minority class, thus leading to an increased recall for the minority class and a reduced recall for the majority class. With a 60:40 distribution similar results are obtained as with a 30:70 or 40:60 distribution for no weighting, while for TF-IDF weighting this pattern is less distinct. Applying a 70:30 distribution again results in a local minimum where most test cases are classified as minority cases, resulting in a high recall for the minority class and low recall for the majority class.

For oversampling, no weighting and TF-IDF weighting also tend to follow a similar pattern, where for TF-IDF this pattern is more distinct. With all training sample distributions, except for a 40:60 distribution, most test cases are predicted to be minority cases, causing the recall of the minority class to outweigh that of the majority class. However, since the F-measure is a harmonic mean, the precision of the minority class leads to a low value. Opposed to the other training sample distributions, with a 40:60 distribution most test cases are predicted to be majority cases. A possible cause for this could be the duplication of highly influential training cases using SMOTE which has been explained above.

When applying a binary weighting to the training set, the F-measures for both under- and oversampling tend to converge as the training sample distribution increases, caused by an increased recall for the minority class and reduced recall for the majority class. This pattern follows similar patterns which have been observed for all other classifiers, which leads to the assumption that a binary weighted neural network is less sensitive for local minima.

With respect to lemmatization and stemming, similar results are observed as with multinomial naive Bayes for the average F-measures. However, underlying these F-measures are different reasons. Even though no general pattern can be discovered regarding recall and precision, the F-measures of the majority class when applying lemmatization outperform the F-measures when applying stemming. Since the F-measures of the minority class are more alike, the average F-measure for lemmatization outperforms that for stemming.

7.9 K-nearest-neighbor

Table 16 holds an overview of the average F-measures obtained using the default implementation of WEKA's IBk classifier (Aha and Kibler, 1991). Based on the results of Peters (2016a) it has been decided to have a fixed amount of 5 neighbors. Due to runtime issues, only the undersampling results have been obtained and oversampling results are omitted in Table 16. In Figure 24 the F-measures of the minority and majority class, using undersampling and lemmatization, are illustrated, while Figure 25 depicts the recall and precision. A complete overview of the illustrations can be found in Appendix 12.8.

Table 16: F-measures using KNN

Resampling technique	**Normalization technique**	**Training sample distribution**	**Weighting technique**		
			No Weighting	Binary	TF-IDF
No Sampling	Lemmatization	16.7:83.3	0.506 (0.519)	0.480 (0.486)	<u>**0.512**</u> (0.522)
	Stemming	16.7:83.3	0.503 (0.515)	0.476 (0.477)	<u>**0.504**</u> (0.511)
Undersampling	Lemmatization	30:70	**0.543** (0.543)	0.538 (0.544)	**0.534** (0.533)
		40:60	0.534	<u>**0.552**</u>	0.505

			(0.535)	(0.550)	(0.514)
		50:50	0.498 (0.513)	0.506 (0.519)	0.454 (0.484)
		60:40	0.435 (0.473)	0.424 (0.466)	0.377 (0.429)
		70:30	0.347 (0.404)	0.320 (0.379)	0.294 (0.350)
	Stemming	30:70	**0.538** (0.538)	0.534 (0.539)	**0.522** (0.522)
		40:60	0.531 (0.532)	**0.543** (0.541)	0.493 (0.505)
		50:50	0.487 (0.504)	0.501 (0.516)	0.439 (0.473)
		60:40	0.426 (0.466)	0.414 (0.458)	0.367 (0.421)
		70:30	0.341 (0.398)	0.305 (0.363)	0.282 (0.335)

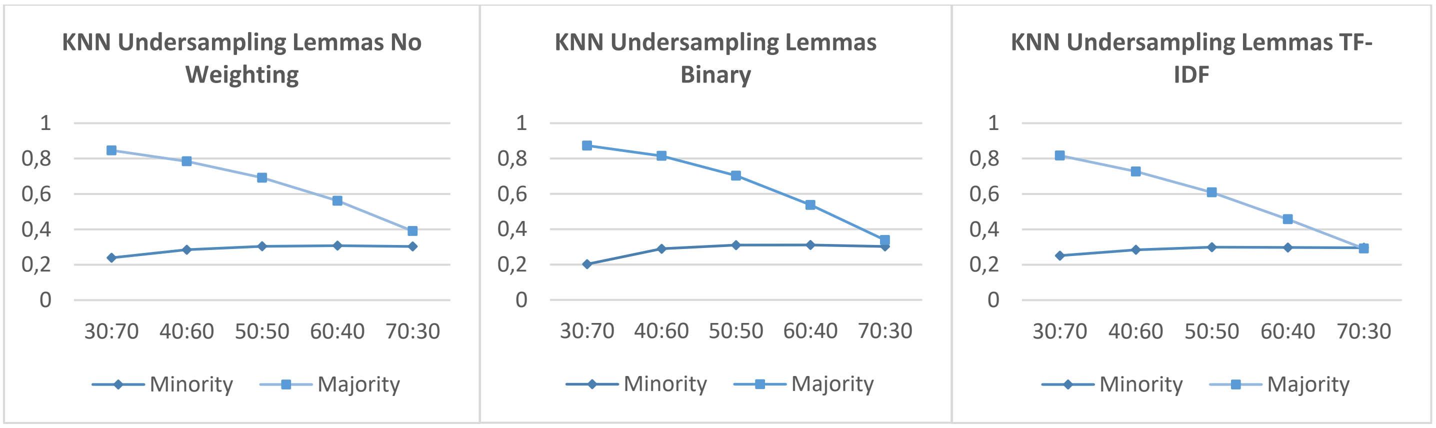


Figure 24: F-measures using K-nearest-neighbor;
X-axis: training sample distribution, Y-axis: F-measure

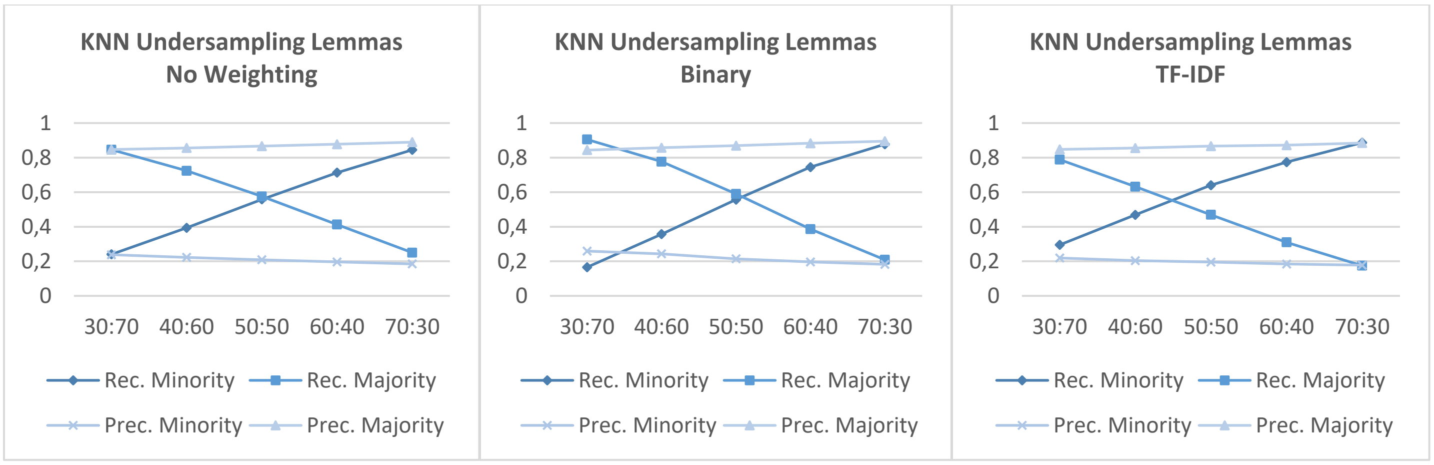


Figure 25: Recall and precision using K-nearest-neighbor;
X-axis: training sample distribution, Y-axis: recall/precision

7.9.1 General results

For KNN, the best classification result is achieved using an undersampled 40:60 distribution with lemmatization and binary weighting. Here, the F-measure of the majority class is decreased by 7.9% with respect to the baseline (89.4%) and the F-measure of the minority class is improved with 17.8%. Overall, resampling a dataset before training a KNN classifier can improve its F-measure up to 4.9%.

7.9.2 Resampling technique

When applying either no weighting or a TF-IDF weighting, the resampling results follow the same pattern as with a decision tree classifier, while applying binary weighting induces the same resampling results as with a logistic regression classifier.

7.9.3 Word normalization technique

With respect to lemmatization and stemming, similar results are observed as with multinomial naive Bayes.

7.9.4 Word weighting technique

Regarding word weighting it can be observed that TF-IDF weighting positively influences the F-measure of an unsampled training set due to a relatively higher recall on the minority class. However, the recall of the majority class is lower in comparison to the other weighting techniques, which becomes more explicit when resampling the training set, leading to a lower average F-measure. When comparing binary and no weighting, it becomes apparent that for training sample distributions up to 50:50, applying binary weighting results in a higher recall for the majority class and a lower recall for the minority class. However, the average F-measure is barely influenced, and no general pattern can be observed.

7.10 Support vector machine

Table 17 holds an overview of the average F-measures obtained using the default implementation of WEKA's SMO classifier (Platt, 1998). Due to runtime issues, the no sampling and oversampling results for binary weighting could not be obtained and here therefore been omitted in Table 17. In Figure 26 the F-measures of the minority and majority class, using undersampling and lemmatization, are illustrated, while Figure 27 depicts the recall and precision. A complete overview of the illustrations can be found in Appendix 12.9.

Table 17: F-measures using support vector machine

Resampling technique	**Normalization technique**	**Training sample distribution**	**Weighting technique**		
			No Weighting	Binary	TF-IDF
No Sampling	Lemmatization	16.7:83.3	**0.454** (0.431)	-	**0.454** (0.431)
	Stemming	16.7:83.3	**0.454** (0.431)	-	**0.454** (0.431)
Undersampling	Lemmatization	30:70	0.460 (0.446)	0.517 (0.525)	0.454 (0.431)
		40:60	**0.568** (0.576)	**0.568** (0.581)	**0.530** (0.546)
		50:50	0.519 (0.537)	0.509 (0.525)	0.514 (0.534)
		60:40	0.365 (0.425)	0.444 (0.493)	0.315 (0.375)
		70:30	0.211 (0.236)	0.263 (0.313)	0.199 (0.215)
	Stemming	30:70	0.457 (0.438)	0.470 (0.455)	0.454 (0.431)
		40:60	**0.556** (0.566)	**0.570** (0.574)	0.504 (0.524)
		50:50	0.506 (0.538)	0.513 (0.531)	**0.505** (0.527)
		60:40	0.362 (0.421)	0.399 (0.453)	0.306 (0.365)
		70:30	0.203 (0.222)	0.250 (0.295)	0.177 (0.174)
Oversampling	Lemmatization	30:70	0.454 (0.431)	-	0.454 (0.431)
		40:60	**0.578** (0.575)	-	**0.589** (0.583)
		50:50	0.508 (0.528)	-	0.543 (0.553)

		60:40	0.421 (0.469)	-	0.450 (0.492)
		70:30	0.263 (0.313)	-	0.265 (0.315)
	Stemming	30:70	0.454 (0.431)	-	0.454 (0.431)
		40:60	**0.567** (0.564)	-	**<u>0.578</u>** (0.574)
		50:50	0.498 (0.520)	-	0.537 (0.548)
		60:40	0.418 (0.466)	-	0.455 (0.464)
		70:30	0.259 (0.308)	-	0.259 (0.307)

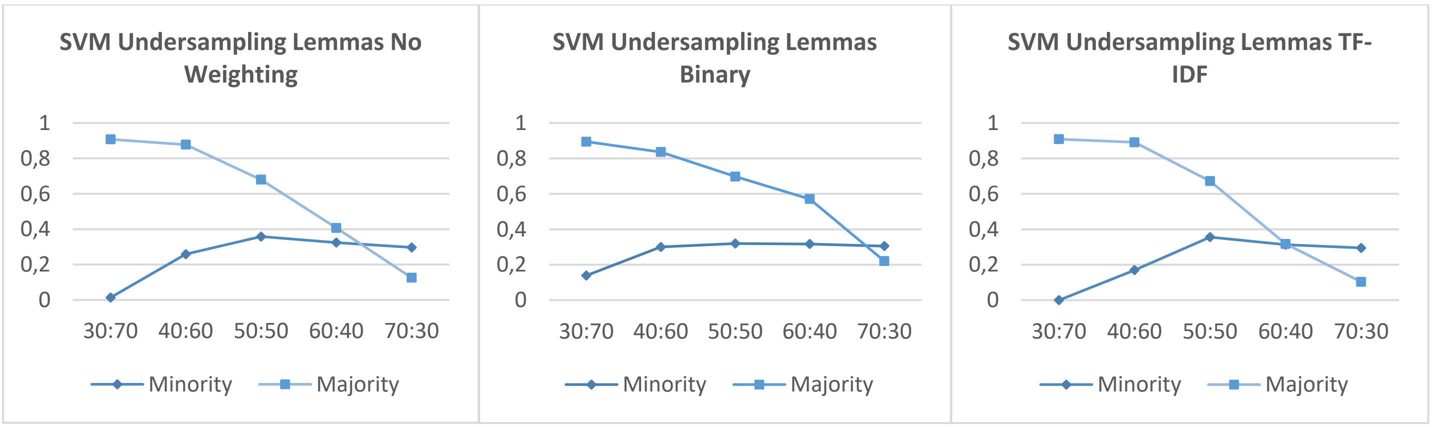


Figure 26: F-measures using support vector machine;
X-axis: training sample distribution, Y-axis: F-measure

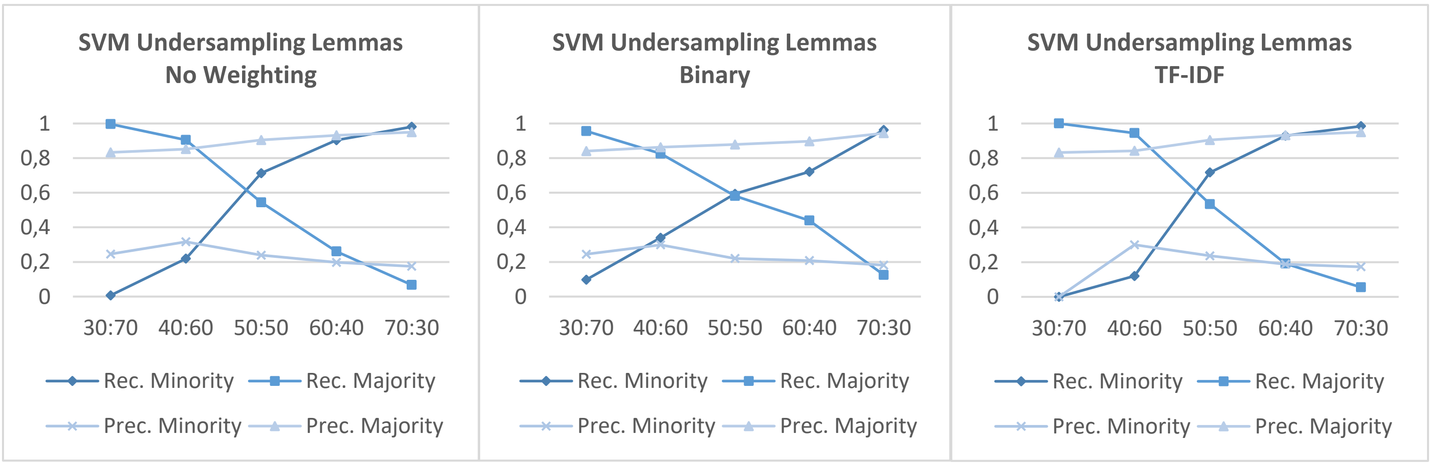


Figure 27: Recall and precision using support vector machine;
X-axis: training sample distribution, Y-axis: recall/precision

7.10.1 General results

Using a support vector machine classifier, the best classification result is achieved for an oversampled 40:60 training sample distribution with lemmatization and TF-IDF weighting. Here, the F-measure of the minority class is improved with 35.4% with respect to the baseline (0%) and the F-measure of the majority class is decreased with 8.6%. By both resampling and weighting a training set, the average F-measure of a SVM classifier can be improved with up to 13.5%.

7.10.2 Resampling technique

Overall, it can be observed that under- and oversampling both follow a similar pattern as with logistic regression, where the average F-measure first increases as the training sample distribution shifts to 40:60 after which it decays. The best classification results for a SVM classifier are therefore consistently obtained using a 40:60 training sample distribution. However, opposed to all other classifiers, a SVM classifier is more positively influenced by oversampling in comparison to undersampling. Initially, a higher recall on the minority class is achieved when using a 40:60 distribution, inducing a higher minority F-measure. This could be caused by SMOTE duplicating highly discriminative cases, which positively influence the training of the classifier. For greater distributions, the recall of the minority class equalizes for both resampling techniques. Besides this first effect, oversampling also induces a lower decay of the majority class recall. When undersampling, most test cases are classified as majority cases up to a 40:60 distribution after which it quickly transitions to most cases being classified as minority cases. However, oversampling causes a more gradual transition, thus resulting in an improved majority recall for all training sample distributions above 40:60.

7.10.3 Word normalization technique

With respect to lemmatization and stemming, similar results are observed as with multinomial naive Bayes.

7.10.4 Word weighting technique

When undersampling, it can be noticed that binary weighting consistently outperforms the other word weighting techniques. A binary weighted SVM classifier induces a more gradual decline of the majority class recall and incline of the minority class recall, therefore resulting in higher average F-measures. Since oversampling results have not been obtained for binary weighting, only TF-IDF and no weighting can be compared. Where for undersampling TF-IDF weighting underperforms in comparison to no weighting, for oversampling the results are reversed. Due to a higher recall on the majority class, a TF-IDF weighted SVM classifier consistently outperforms an unweighted SVM classifier.

7.11 AdaBoost

Table 18 holds an overview of the average F-measures obtained using the default implementation of WEKA's AdaBoostM1 classifier (Freund and Schapire, 1996). In Figure 28 the F-measures of the minority and majority class, using undersampling and lemmatization, are illustrated, while Figure 29 depicts the recall and precision. A complete overview of the illustrations can be found in Appendix 12.10.

Table 18: F-measures using adaboost

Resampling technique	**Normalization technique**	**Training sample distribution**	**Weighting technique**		
			No Weighting	Binary	TF-IDF
No Sampling	Lemmatization	16.7:83.3	0.454 (0.431)	0.454 (0.431)	0.454 (0.431
	Stemming	16.7:83.3	0.454 (0.431)	0.454 (0.431)	0.454 (0.431)
Undersampling	Lemmatization	30:70	0.463 (0.449)	0.478 (0.474)	0.471 (0.463)
		40:60	**0.544** (0.564)	**<u>0.556</u>** (0.577)	**0.539** (0.567)
		50:50	0.507 (0.527)	0.515 (0.532)	0.506 (0.536)
		60:40	0.410 (0.462)	0.408 (0.549)	0.393 (0.449)
		70:30	0.282 (0.337)	0.250 (0.298)	0.264 (0.316)
	Stemming	30:70	0.467 (0.454)	0.470 (0.455)	0.472 (0.462)
		40:60	**0.535** (0.555)	**<u>0.538</u>** (0.559)	**0.516** (0.527)
		50:50	0.495 (0.518)	0.498 (0.520)	0.492 (0.517)
		60:40	0.404 (0.453)	0.403 (0.454)	0.340 (0.451)
		70:30	0.158 (0.134)	0.164 (0.148)	0.160 (0.138)
Oversampling	Lemmatization	30:70	0.454 (0.431)	0.454 (0.431)	0.531 (0.542)
		40:60	**<u>0.553</u>** (0.553)	**0.550** (0.553)	**0.550** (0.553)
		50:50	0.513 (0.529)	0.515 (0.531)	0.537 (0.541)
		60:40	0.458 (0.494)	0.454 (0.589)	0.448 (0.501)

		70:30	0.347 (0.407)	0.351 (0.410)	0.338 (0.408)
	Stemming	30:70	0.454 (0.431)	0.454 (0.531)	0.531 (0.546)
		40:60	**0.535** (0.536)	0.494 (0.490)	**<u>0.551</u>** (0.556)
		50:50	0.483 (0.506)	**0.497** (0.519)	0.534 (0.536)
		60:40	0.434 (0.475)	0.416 (0.462)	0.417 (0.461)
		70:30	0.374 (0.430)	0.327 (0.386)	0.284 (0.344)

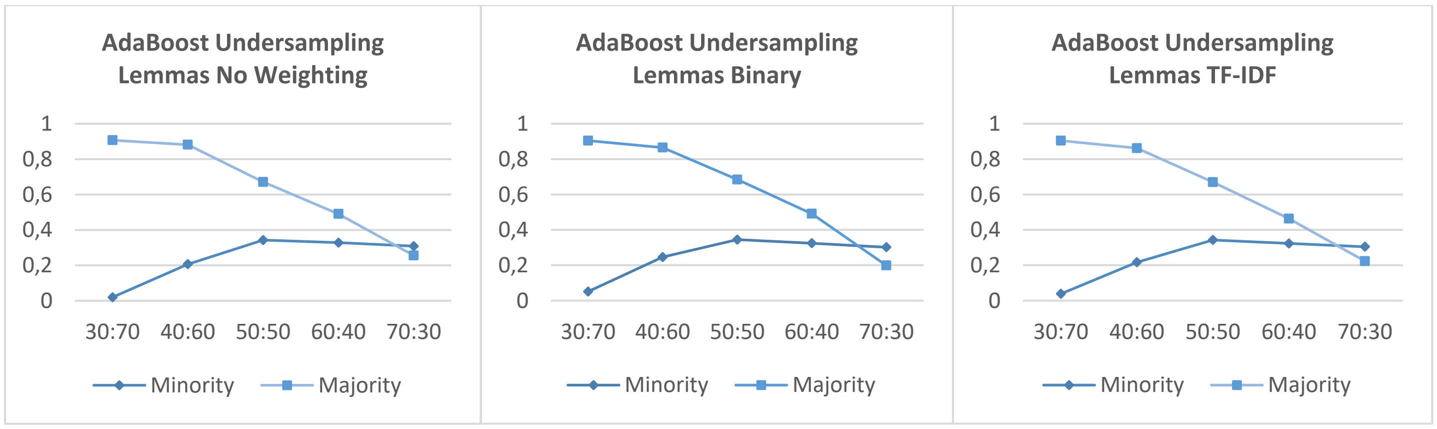


Figure 28: F-measures using adaboost;
X-axis: training sample distribution; Y-axis: F-measure

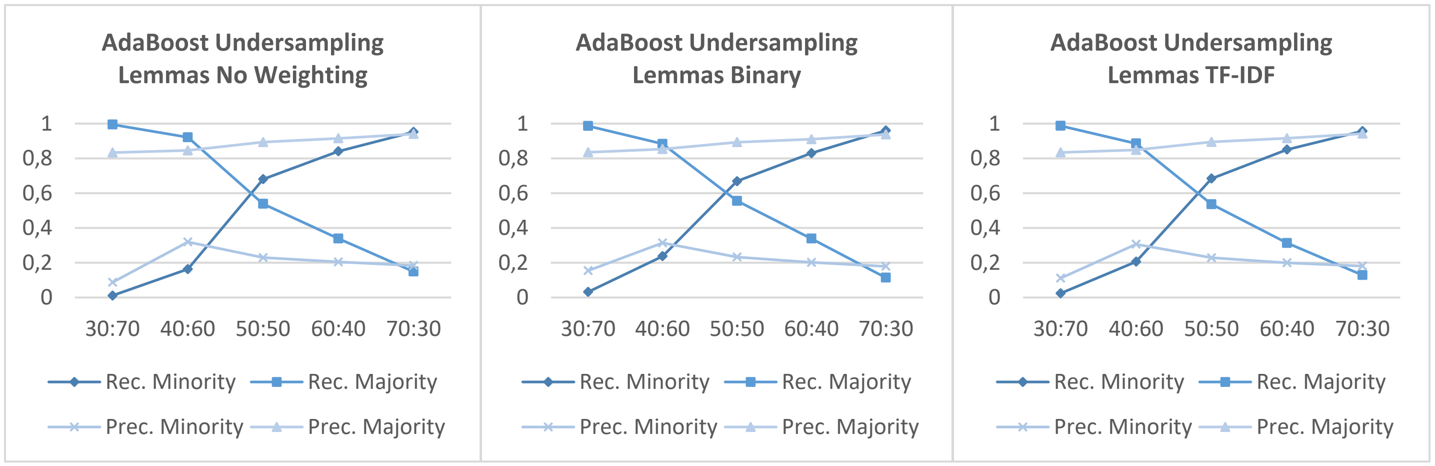


Figure 29: Recall and precision using adaboost;
X-axis: training sample distribution; Y-axis: recall/precision

7.11.1 General results

For AdaBoost, the best classification result is achieved using an undersampled 40:60 distribution with lemmatization and binary weighting. Here, the F-measure of the majority class is decreased by 4.4% with respect to the baseline (90.9%) and the F-measure of the minority class is improved with 24.6%. Overall, resampling a dataset before training an AdaBoost classifier can improve its F-measure up to 10.2%.

7.11.2 Resampling technique

With respect to undersampling, the classification results obtained using an AdaBoost classifier tend follow the same pattern as those obtained using an SVM classifier. For oversampling, applying both binary and no weighting triggers the same response for an AdaBoost classifier, where the recall of the minority class ameliorates as the training sample distribution shifts up to 40:60, after which its increase stabilizes again. The decay of the majority class recall is also larger up to a 40:60 distribution, however this decay is less influential to the macro-averaged F-measure than the increase in recall of the minority class. An underlying reason for this phenomena could be that highly discriminative cases have been duplicated using SMOTE which positively influences the overall recall of an AdaBoost classifier, just as with oversampling an SVM classifier. In general, oversampling has a greater influence on the classification performance as compared to undersampling, thereby resembling the results found for a decision tree classifier. Since AdaBoost combines a collection of small decision trees using majority voting, this outcome is therefore as expected.

7.11.3 Word normalization technique

With respect to lemmatization and stemming, similar results are observed as with multinomial naive Bayes.

7.11.4 Word weighting technique

When undersampling, all weighting techniques influence an AdaBoost classifier in the same manner. Increasing the training sample distribution up to 40:60 only influences the recall and precision of the minority class, while the majority class holds similar classification results. Starting from a 50:50 distribution, the recall of the majority class rapidly diminishes thereby decreasing the majority class F-measure. Due to the harmonic nature of the F-measure, the minority F-measure is left unaffected by the increase in minority recall. For oversampling, both binary and no weighting have a similar effect on the classifier, with a sudden spike of the minority and majority recall at a 40:60 distribution. Applying TF-IDF weighting, however, results in a more gradual transition of both the minority and majority recall, even though this cannot be observed in the macro-averaged F-measures.

8 Discussion

This chapter discusses the findings mentioned in the results chapter and critically assesses them with respect to considerations made during this research and the results from previous researches.

8.1 Evaluating the overall results

During this research a total of 9 classifiers have been evaluated with respect to their behavior to resampling, word normalization and word weighting. An overview of the individual results has been combined in Table 19, in which for each classifier the baseline is compared to the optimal training set setup. This optimal setup is included in between brackets, where the first letter represents the resampling type, the number the percentage of minority cases, and the last letter the weighting type.

Table 19: Overview individual classifier results

Classifier	**Baseline**	**Optimal**	**Difference**
Multinomial naive Bayes	0.572	0.590 (U30B)	0.018
Logistic regression	0.466	0.594 (U40N)	0.128
Decision tree	0.527	0.560 (U30T)	0.033
Multivariate naive Bayes	0.560	0.586 (N16B)	0.026
Association rule	0.456	0.585 (U40N)	0.129
Neural network	0.454	0.564 (U30B)	0.110
K-nearest-neighbor	0.503	0.552 (U40B)	0.049
Support vector machine	0.454	0.589 (O40T)	0.135
AdaBoost	0.454	0.556 (U40B)	0.102

When evaluating the observations of each individual classification algorithm, it becomes apparent that the difference in optimized performance between the best classifier (i.e. Logistic regression) and the worst classifier (i.e. K-nearest-neighbor) is only minor with 4.2 percentage points (i.e. pp). Four classifiers hold the best optimized classification performance within a range of 1pp, namely the logistic regression, multinomial naive Bayes, multivariate naive Bayes, and association rule classifiers. Even though the differences between the individual baseline and optimal classification results vary in size, this minor difference in optimal performance suggests that a classifier can only be improved up to a certain extent. This would thus imply that, after optimizing, the selection of an appropriate classification algorithm should depend less upon performance, but more on other metrics (e.g. runtime). In our results, the macro-averaged F-measures do not exceed 0.594, which results from a low performance on the minority class. Underlying this relatively low overall performance could be a variety of reasons:

- Since employees at DLOC and LMIO were merely interested in whether a complaint will be withdrawn or not, individual withdrawal reasons were not taken into account. When analyzing the reasons for withdrawal it became apparent that the majority of the withdrawals (i.e. 4.901 of the 8.609) result from either refunding the money or receiving of the ordered goods. These two reasons for withdrawal are caused by human acting which is difficult to predict using a static text field. Because the majority of the minority class consists of cases which were withdrawn based on human acting, it makes sense that the performance on the minority class is rather disappointing.
- During the data understanding phase as explained in chapter 6.1 the features used in the free-text field for both withdrawn and not withdrawn complaints showed a high resemblance. When, for example, warning the police for a possible fraudster, which is not a valid complaint, words like “marktplaats” (online trading website), “oplichter” (fraudster), and “product” (idem) are often used. Such explanative features are also used in a complaint which has not been withdrawn. Comparing a subset of individual complaints revealed that it is possible that the free-text fields do not contain enough information to be distinct, thereby reducing the overall performance of a classifier trained on this dataset using features resulting from a bag-of-words approach.
- Since in this research it has been opted to use the 100 most occurring terms, which are used in both withdrawn and not withdrawn complaints, the overall performance of the classifiers could be reduced compared to when the more distinctive features would be selected using a feature selection algorithm.

8.2 Evaluating the data alteration results

8.2.1 Resampling

With respect to resampling, each individual classification technique follows a similar trend. When increasing the training sample distribution, the precision of the minority class slightly decreases, while the recall of the minority class strongly increases. The majority class follows a similar pattern where the precision of the majority class slightly increases, while the recall of the majority class strongly decreases. Even though resampling is intended to increase the minority performance without detriment of the majority performance, the observed pattern can be explained. During the training phase a classifier will, at first, become less inclined to automatically label a test case as the majority class as the skewness is lifted, after which it will be inclined towards the minority class as it becomes overly present in the training set. The rate at which the recall and precision change differs per classification technique, however, the precision of the minority class does, independent of the training sample distribution, remain centered around 30% for each classification technique. In a similar way, the precision of the majority class remains centered around 90%.

When further evaluating the above described pattern, it becomes clear that, in our results, increasing the percentage of minority cases can only improve the F-measure of the minority class up to a certain extent due to its harmonic nature. A low precision will always overshadow the increase in recall, thereby restricting its positive influence on the F-measure. For the majority class, the F-measure only decreases due to the decline in recall, thus nullifying the influence of the high precision. Combining both patterns results in the conclusion that an optimal resampling percentage should thus be at a level where the initial increase in minority performance neutralizes the decrease in majority performance. This initial increase is the largest when there are less minority cases, which implies that the resampling percentage should be in favor of the majority class, resembling the initial minority/majority ratio. In Table 19, all optimal results are achieved using either a 30/70 or a 40/60 distribution, thereby supporting above conclusion.

Comparing both under- and oversampling results in general it is found that undersampling often slightly outperforms oversampling, which is also shown in Table 19 where 7 of the 9 optimal results were achieved using undersampling. In his initial research, however, Peters (2016a) concluded oversampling outperforms undersampling, same as Japkowicz and Stephen (2002) did in their research. A possible explanation for this difference could lie in the dataset and features used in this research. When oversampling training cases with little distinctive features, a classification algorithm is presented with just more cases instead of more distinctive cases. Due to the higher absolute number of cases, a classifier could be more inclined towards the majority class as would be with undersampling, where the same distribution but less cases are presented. The difference in performance between under- and oversampling, however, was found to be minor for most training setups and should therefore not be used as the only metric for deciding on which resampling technique to use.

8.2.2 Word weighting

In Chapter 7, the influence of word weighting with respect to each individual classifier has been discussed. To evaluate the overall influence of word weighting, these individual results are combined in Table 20. Here, the weighting technique with the highest absolute number of cases in which it resulted in the best performance for each resampling technique is marked with an X. When multiple weighting techniques perform best on the same amount of cases they are both marked with an X.

Looking at the table it becomes apparent that for most classifiers, the right choice of weighting technique can outperform an unweighted baseline. Which technique to use depends upon the classification technique, however, binary weighting often outperforms TF-IDF weighting. This finding could imply that merely the presence of a term is enough for a classifier to be based upon, which is consistent with earlier findings with respect to SVM and naive Bayes classifiers (Jurafsky & Martin, 2014; Pang et al. 2002; Schneider, 2004).

Table 20: Overview word weighting influence

Classification technique	**Resampling technique**	**Weighting technique**		
		No weighting	Binary	TF-IDF
Multinomial naive Bayes	No sampling	X		
	Undersampling	X	X	
	Oversampling		X	
Logistic regression	No sampling	X		
	Undersampling		X	
	Oversampling		X	
Decision tree	No sampling	X		
	Undersampling			X
	Oversampling		X	
Multivariate naive Bayes	No sampling		X	
	Undersampling		X	
	Oversampling		X	
Association rule	No sampling		X	X
	Undersampling	X		
	Oversampling			X
Neural network	No sampling		X	
	Undersampling		X	
	Oversampling		X	
K-nearest-neighbor	No sampling			X
	Undersampling	X		
Support vector machine	No sampling	X		X
	Undersampling		X	
	Oversampling			X
AdaBoost	No sampling	X	X	X
	Undersampling		X	

	Oversampling			X

8.2.3 Word normalization

When comparing the classification performance with respect to word normalization only one conclusion can be drawn, namely that lemmatization poses a better effect than stemming. In Table 21, the difference in performance using either stemming or lemmatization has been compared for all classifiers under the optimal setup. As can be seen from the table, all classifiers perform better using lemmatization, however, the difference in performance is only minor, ranging from 0.5 to 1.8pp. Overall, with a few exceptions, lemmatization has little but consistently improved the recall and precision of both the minority and majority class.

Table 21: Overview word normalization influence

Classifier	**Stemming**	**Lemmatization**	**Difference**
Multinomial naive Bayes	0.584	0.590	0.006
Logistic regression	0.583	0.594	0.011
Decision tree	0.549	0.560	0.011
Multivariate naive Bayes	0.581	0.586	0.005
Association rule	0.574	0.585	0.011
Neural network	0.551	0.564	0.013
K-nearest-neighbor	0.543	0.552	0.009
Support vector machine	0.578	0.589	0.011
AdaBoost	0.538	0.556	0.018

8.3 Evaluating the additional results

In chapter 8.1, three possible reasons have been mentioned with respect to the relatively low classification performance on minority class. To initially examine these possible causes, follow-up experiments have been set up in which the influence of each cause is independently verified. Furthermore, an experiment has been executed regarding the probabilistic classifiers as the findings in this research did not match the prescribed literature.

8.3.1 The performance of probabilistic classifiers

When comparing findings regarding individual classifier performance with previous research, it can be observed that probabilistic classifiers perform better on this dataset. In their research, Yang and Liu (1999) found logistic regression, SVM, and KNN to significantly outperform neural networks and naive Bayes with respect to the macro-averaged F-measure on the Reuters-21578 corpus. Even though it is based upon the micro-averaged F-measure, the result comparison of Sebastiani (2002) also showed the lower performance of probabilistic classifiers on the skewed Reuters corpus in comparison to the other classifiers in his research. Here it should be noted that the Reuters corpus is a multi-labeled dataset instead of binary, and that the classifiers in the overview of Sebastiani (2002) have not been improved using resampling. However, when comparing the unsampled results, the probabilistic classifiers still outperform all other classifiers in this research including logistic regression and association rules. Looking at the probabilistic classifiers individually, it was observed that the difference in performance between an unsampled and a resampled classifier was negligible. Combining this with the skewed characteristic of the dataset led to the assumption that on a highly skewed dataset, determining the posterior probability in a naive Bayes classifier is mainly influenced by the class likelihood and less by the class priori.

To support this assumption, the class likelihood estimates of an unsampled MNB classifier using stemming and no weighting have been included in Appendix 10.11. The table shows the likelihood of the most present feature to be only 0.041 for the majority class and 0.038 for the minority class. In the prediction of a class using a naive Bayesian classifier the priori is multiplied with the respective likelihood of the present features. When, for example, 5 features are contained in the test set, this would in the best possible case (i.e. 0.041^5) result in a likelihood multiplication of 0.000000115 (i.e. 1.15E-7) for the majority class. Multiplying this with the priori of 0.83 would only slightly alter this outcome to 9.6E-8. This same principle applies to the minority class, where due to the low value resulting from the likelihood multiplication, the priori is of minor influence when calculating the posterior probability (i.e. 7.9E-8 to 1.4E-8). The influence of the priori is thus determined by the amount of features present in the test set. When the test set contains only one feature, the influence of the priori is rather large, however, when the amount of features present in the test set increases, this influence gradually decreases. A lower priori influence makes a probabilistic classifier more dependent upon the class likelihoods, thereby uncovering the predictive power of the used features. Above observations could imply that [1] the predictive power of a probabilistic classifier on textual data depends upon the length of cases in the test set, or [2] the chosen features in this research cannot be used for predicting the class label. To either confirm of disconfirm this, an experiment has been set up in which the class predictions were evaluated with respect to the total amount of features in the test set which have been used for building the classifier. For this experiment, the multinomial naive Bayes classifier has been trained according to the baseline and framework as described in Chapter 7. The results of this experiment are contained in Table 22 and depicted in Figure 30 and Figure 31.

Table 22: Results per feature set using multinomial naive Bayes

#Features in test set	**Set size**	**Recall minority**	**Recall majority**	**Precision minority**	**Precision majority**	**F-measure minority**	**F-measure majority**
0-9	1529.2	0.068	0.965	0.313	0.825	0.110	0.889
10-19	1921.8	0.273	0.884	0.331	0.856	0.298	0.870
20-29	1016.4	0.343	0.844	0.298	0.871	0.318	0.857
30-39	459.9	0.443	0.827	0.309	0.894	0.361	0.859
40+	211.3	0.482	0.819	0.281	0.920	0.348	0.866

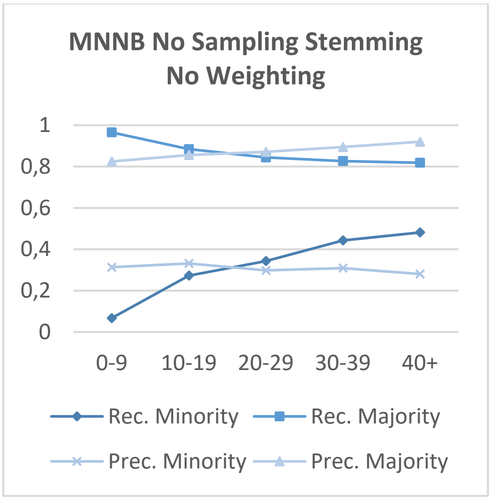


Figure 30: Recall and precision using multinomial naive Bayes; X-axis: #features in test set, Y-axis: recall/precision

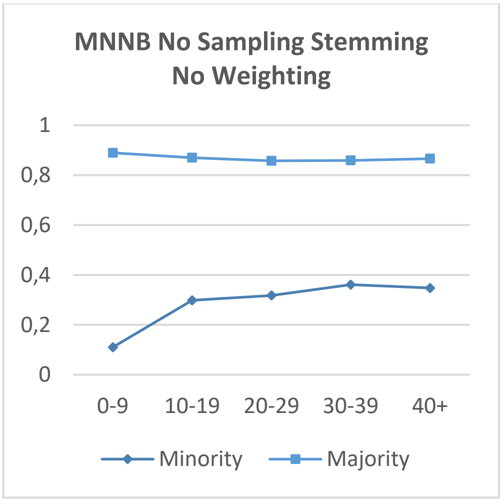


Figure 31: F-measure using multinomial naive Bayes; X-axis: #features in test set, Y-axis: F-measure

Evaluating above results reveals a clear relationship between the predictive accuracy of a classifier and the amount of features used in the test set. Using less features causes a multinomial naive Bayes classifier to predict more test cases as the majority class compared to using more features. This observation supports the assumption that the influence of the priori decreases as more features are present in the test set. Furthermore, by increasing the amount of features in the test set, the average F-measure increases due to a higher recall for the minority class. Figure 31 suggests, however, that this effect follows a logarithmic equation, thereby lessening the impact on the minority recall as more features are used in the test set, which is confirmed by an R^2 value of 0.992 for the equation $y = 0.2572 ln(x) + 0.0753$. To ensure this result is not induced by the fortunate combination of outliers, the standard deviations of all feature set sizes have been evaluated through Figure 32, which confirms the logarithmic nature of the minority recall. Combining all findings from this experiment, it can be concluded that the predictive power of a probabilistic classifier on a skewed dataset depends upon the amount of features used in the test set. Since in this research the 100 most common features have been used, the amount of features used in the test set are likely to be correlated with the length of the test set, thereby supporting the hypothesis that the predictive power of a classifier on textual data depends upon the length of the cases in the test set.

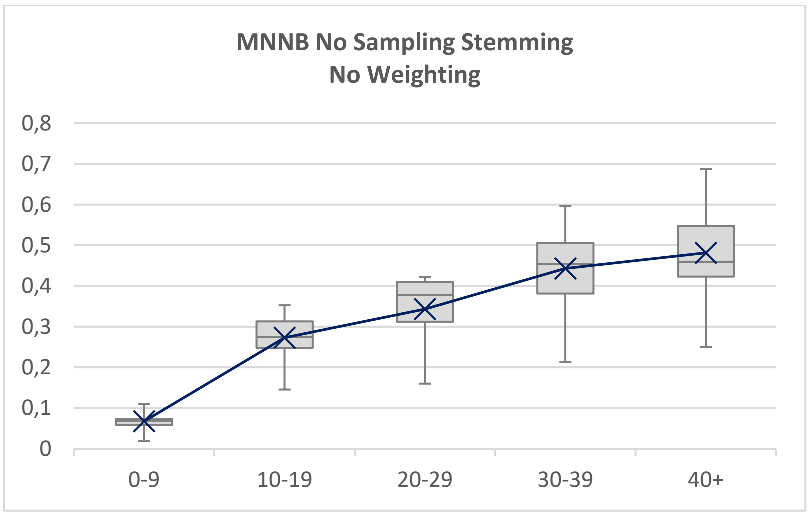


Figure 32: Minority recall using multinomial naive Bayes;
X-axis: #features in test set, Y-axis: minority recall

8.3.2 The influence of feature selection

In chapter 8.2.1 it was mentioned that independent of the training sample distribution, the precision of the minority class remains fixed around 30%. This implies that, even though the 100 most common features contain more information than simply classifying all test cases as either the minority or majority class, the information richness is restricted. To examine whether feature selection could be used to overcome this limitation in information richness, an experiment following the same training procedure as above has been set up in which bi-normal separation (Forman, 2003) is used to select the most informative features. Since it is unknown whether a total of 100 selected features accurately represents the classification performance, as was with selecting the most common features, the number of features used for training the classifier range from 100 to 5000. The results of this experiment are contained in Table 23 and depicted in Figure 33 and Figure 34.

Table 23: Results using multinomial naive Bayes with bi-normal separation

#Features in training set	Recall minority	Recall majority	Precision minority	Precision majority	F-measure minority	F-measure majority
100	0.033	0.989	0.384	0.836	0.060	0.906
1000	0.221	0.879	0.275	0.849	0.241	0.863
5000	0.440	0.799	0.306	0.876	0.360	0.835

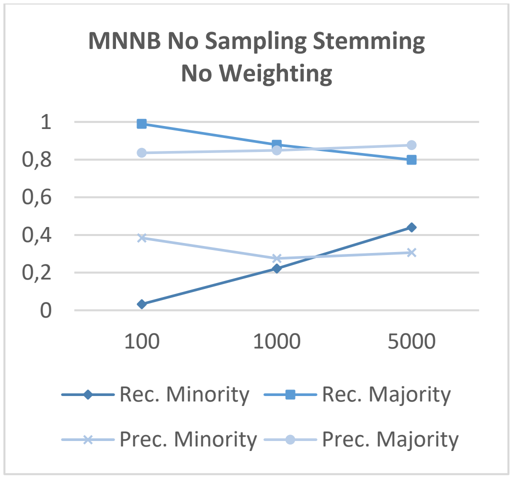


Figure 33: Recall and precision using multinomial naive Bayes; X-axis: #features in training set, Y-axis: recall/precision

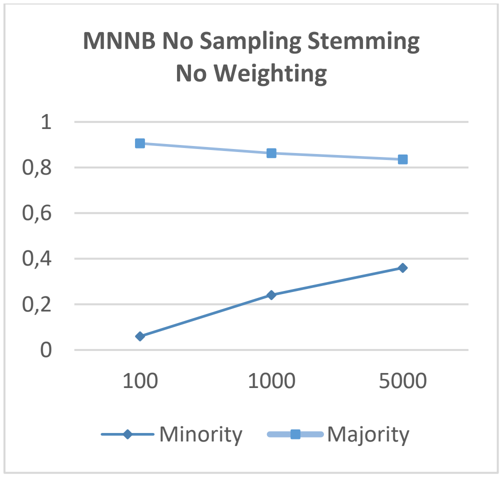


Figure 34: F-measures using multinomial naive Bayes; X-axis: #features in training set, Y-axis: F-measure

When exploring above results, it becomes apparent that increasing the amount of features used for training the classifier induces an increase in test cases to be classified as the minority class (i.e. minority recall). Since the recall of the minority class cannot extend 1, the seemingly linear minority recall trend in Figure 33 must level and is likely to be a logarithmic trend as in Figure 32. Based on the current results, however, it is hard to determine at which rate the minority recall levels, as could be in Figure 32.

Interestingly, the minority recall begins at a relatively low value (0.033) when using 100 features selected with bi-normal separation compared to the minority recall using the 100 most occurring features (0.241). This observation can be explained through the way feature selection operates. By comparing the ratio of a feature occurring in a minority case to that of it occurring in a majority case, features can be ordered based on their distinctiveness (i.e. a high comparison ratio). A feature occurring in 10% of the minority cases and 1% of the majority cases is thus more informative than a feature occurring in 5% of both the minority and majority cases. This comparison ratio, however, does pose a downside, namely that it does not take into account the number of absolute occurrences of a feature. When, for example, a feature occurs in 10 minority cases and 0 majority cases, it is considered to be highly informative and shall thus be selected. With a total of 1000 minority cases, however, this feature discerns only a small portion of the training set, and a selected feature occurring in 150 minority and 25 majority cases would have been better. In this research, this downside has been tried to overcome using a minimum of 25 occurrences.

Given that the selected features are highly distinctive, the low minority recall using 100 features combined with the followed increase when using more features suggests a reduced influence of the priori which, as was discussed earlier, implies few features are used for testing when training the classifier with 100 selected features. Combining this with above example leads to the conclusion that the distinctive features selected using bi-normal separation do not cover a wide spectrum of the test cases. Since the most distinctive features occur in only a limited amount of complaints, the assumption is supported that the features in the dataset do not contain enough information to be used for making a distinction on whether a complaint will be withdrawn or not.

In order to conduct a fair comparison between training on the most occurring features and feature selection, the performance of a multinomial naive Bayes classifier on the respectively 100, 1000, and 5000 most occurring features also had to be tested according to the same training procedure. The results of this experiment are contained in Table 24 and depicted in Figure 35 and Figure 36.

Table 24: Results using multinomial naive Bayes with most common feature selection

#Features in training set	Recall minority	Recall majority	Precision minority	Precision majority	F-measure minority	F-measure majority
100	0.241	0.892	0.314	0.854	0.272	0.872
1000	0.412	0.804	0.299	0.872	0.346	0.836
5000	0.474	0.780	0.303	0.881	0.369	0.827

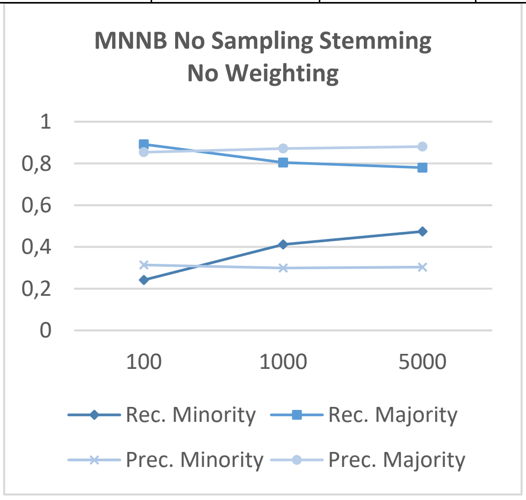


Figure 35: Recall and precision using multinomial naive Bayes; X-axis: #features in training set, Y-axis: recall/precision

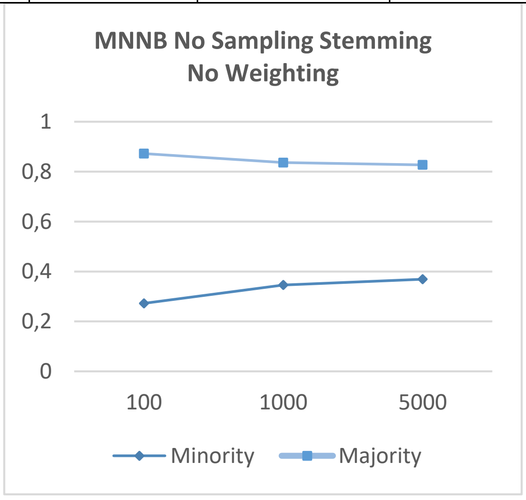


Figure 36: F-measures using multinomial naive Bayes; X-axis: #features in training set, Y-axis: F-measure

As can be seen from Figure 35, increasing the amount of most common features for training also induces a logarithmic increase in the minority recall and thereby the minority F-measure. Since the average F-measure only increases by 2.6pp when using 5000 instead of 100 features, while the training time increases from 15 minutes to 24 hours, it was, as explained in Chapter 7.1, chosen to use only 100 features. With the results found in this research it should, however, be kept in mind that increasing the number of features used for training will increase the performance on the minority class for a multinomial naive Bayes classifier. Further research is required to conclude whether increasing the amount of features used also positively influences the minority performance for other classifiers.

Comparing the results of the two feature selection methods, it can be observed that for 100 features (-8.9pp) and 1000 features (-4.9pp), using bi-normal separation as the feature selection metric only reduces the performance of multinomial naive Bayes classifier, while for 5000 features there is no difference in performance. Combining this with above conclusion that the distinctive features in this dataset cover only a small spectrum of the test cases, it can be concluded that when training a multinomial naive Bayes classifier on a dataset with little distinctive features the best feature selection metric is to use the most common features. Further research is required to conclude whether this finding also applies to other classifiers.

8.3.3 The influence of additional data cleansing

As explained in Chapter 8.1, the reason for withdrawal could play a role in the relatively low classification performance on the minority class. To see whether cases in which human acting plays an active role influence the classification performance, an experiment has been set up in which the 4901 minority cases withdrawn due to either refunding the money or receiving of the ordered goods were removed from the dataset. This left 3708 minority cases remaining, thereby increasing the skewness of the dataset to 8.0%. The followed experiment procedure was similar to above experiments, with the number of features used for training the classifier ranging from 100 to 5000. The results of this experiment are contained in Table 25 and depicted in Figure 37 and Figure 38.

Table 25: Results using multinomial naive Bayes on the reduced dataset

#Features in training set	**Recall minority**	**Recall majority**	**Precision minority**	**Precision majority**	**F-measure minority**	**F-measure majority**
100	0.178	0.947	0.221	0.930	0.200	0.938
1000	0.463	0.870	0.232	0.950	0.308	0.908
5000	0.530	0.863	0.245	0.955	0.334	0.906

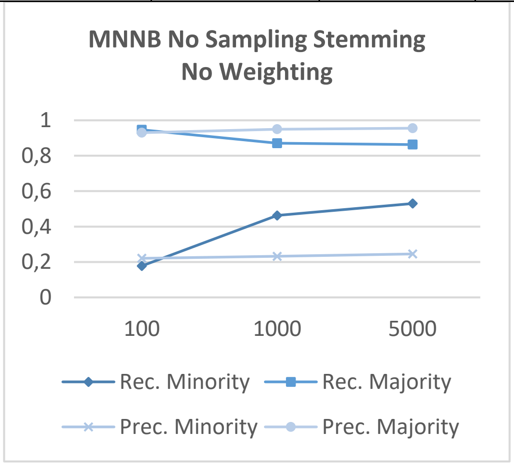


Figure 37: Recall and precision using multinomial naive Bayes; X-axis: #features in training set, Y-axis: recall/precision

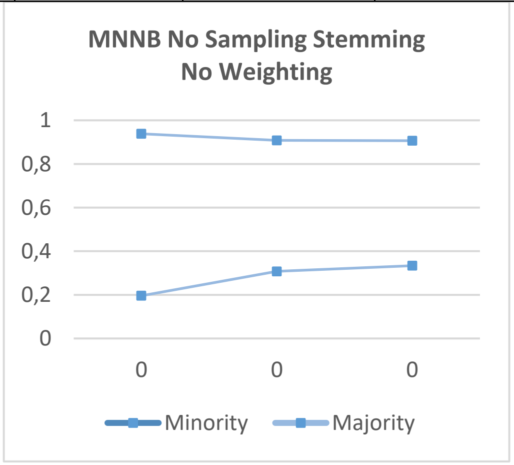


Figure 38: F-measures using multinomial naive Bayes; X-axis: #features in training set, Y-axis: F-measure

By removing the minority cases in which human acting plays an active role, the minority F-measure is slightly reduced, while the majority F-measure is increased. Underlying this pattern is an increased majority recall and precision combined with a decreased minority precision as compared to Figure 35. Even though the informative value in these specific cases helps in more accurately classifying minority cases, its diminishing effect on the majority performance is greater. This leads to the conclusion that the information in these specific minority cases resembles that of the majority cases, thereby supporting the assumption that the free-text field in this dataset does not contain enough information to accurately distinct withdrawn from not withdrawn cases. Overall, when training a multinomial naive Bayes classifier on the reduced dataset with either 1000 of 5000 features, the classification performance can be improved up to 2.2pp.

9 Conclusion

Due to the rising trend in online trade fraud, the Dutch police has established a police unit responsible for solving online trade fraud cases, LMIO, which cooperates with an administrative police unit that handles incoming data, DLOC (Inspectie Veiligheid en Justitie, 2015). Since the police does not have enough manpower to turn each complaint into a case, it is important that a distinction is made as early as possible between complaints worth investigating and those not worth investigating, so that resources are not wasted. This research has focused on whether machine learning techniques can be utilized for this purpose and thereby (partly) automate the handling of online trade fraud complaints. The above statements have been combined in a main research question:

"How can the handling of online trade fraud complaints be (partly) automated?"

In order to answer this main research question, a series of sub-questions were formed as described in Chapter 1. In this chapter, these sub-questions will first be individually answered using the conducted literature studies and data analysis, after which an answer for the main research question will be formulated.

9.1 Answering the sub-questions

9.1.1 SQ1: What are the characteristics of complaint data?

In order to answer this sub-question, the dataset provided by the Dutch National Police has been extensively reviewed as described in chapter 6.4. The dataset consists of 51.386 complaints, which have all been manually labelled by employees at DLOC on whether a complaint has been withdrawn or not and for what reason. Each complaint consists of personal details of the complainant, as well as the personal details of the fraudster insofar they are known. Next to these personal details, each complaint contains information regarding the actual fraud (e.g. type of product, commerce site details, payment method, product price) combined with a free-text field in which the complainant's story that led to the complaint is included. Characteristic of this dataset is that the vocabulary used for describing a complaint in this free-text field are alike, independent of whether it has been withdrawn or not.

Each complaint contains a total of 60 attributes, including the binary class label stating whether a complaint has been withdrawn or not. In total, 8.609 (16.7%) entries have been labelled as withdrawn. Due to the offset ratio of withdrawn to not withdrawn complaints the online trade fraud complaints dataset is to be considered as a skewed dataset, which involves one class being overly present in the dataset.

9.1.2 SQ2: Which steps in the process of handling a complaint can be (partly) automated given the current data availability?

For answering sub-question 2 and uncovering the relevant characteristics for handling an online trade fraud complaint, interviews were conducted at both DLOC and LMIO, as described in Chapter 4.3 and 5. Next to these interviews, a literature study was conducted which was mostly based upon the work by Latukolan and van Ginkel (2016). The result was a workflow for the most important processes at both DLOC and LMIO. During the interviews, these processes were further elaborated upon, and it became obvious that a tailored approach for each complaint restricts the process at LMIO from being (partly) automated. Therefore the focus of this research and correspondingly this sub-question shifted entirely to the process of handling a complaint at DLOC. Based on the available dataset and feedback during the interviews, it was opted to emphasize this research on the registering of a complaint. During the registering of a complaint, two characteristics are of importance: the type of a complaint, and the completeness of a complaint. Since research by Bex et al. (2016) focuses on improving the completeness of a complaint, it was chosen in consultation to direct this research towards predicting the type of a complaint, specifically whether a complaint will be withdrawn or not. Through this prediction, more important complaints (i.e. those that will not be withdrawn) can be dealt with earlier, thereby increasing the processing speed of those complaints at DLOC. This also improves the process at LMIO, since feedback regarding a complaint withdrawal is received faster, which will reduce the time spent on complaints not worth investigating.

9.1.3 SQ3: Can machine learning techniques be utilized for complaint handling?

To answer this sub-question, three literature studies were performed. The first literature study has focused on the usage of machine learning techniques in crime analysis in general and with respect to complaint handling. Resulting from this literature study was the finding that no specific set of techniques exists which is more often used than others for crime analysis in general. Furthermore, it became apparent that even though machine learning has been used for other purposes (Gupta et al., 2016), it is mostly applied to assist with fraud detection (Sharma & Panigrahi, 2012). Other than previous research by Peters (2016a) no literature on the usage of machine learning techniques in the context of online trade fraud complaints could be found. A second conducted literature study was aimed at the textual nature of the dataset as mentioned under sub-question 1. An overview of machine learning techniques used for classifying textual data has been given by Sebastiani (2002), who concluded that Ensembling based (e.g. AdaBoost), SVM, logistic regression, association rule (e.g. RIPPER), KNN, decision tree, and neural network classifiers all have a high performance on textual data, while probabilistic classifiers have a lower performance. The third literature study examined the influence of a skewed dataset on the performance of machine learning techniques. It was found that standard machine learning techniques are designed to pay more attention to majority classes, and thereby often perform poorly on a skewed dataset. By performing data alterations to the training set, this problem could be solved (Japkowicz and Stephen, 2002; Phua et al., 2004). Specifically, three data alteration techniques were found to positively influence the performance of a machine learning technique on a skewed dataset: resampling, word weighting, and word normalization (Japkowicz and Stephen, 2002; Schneider, 2004; Van den Bosch et al., 2007). By combining the results from all literature studies it was concluded that the machine learning techniques described by Sebastiani (2002) cover the spectrum of techniques used for classifying textual, skewed, and criminal data.

In his research, Peters (2016a) obtained initial promising results when exploring the use of machine learning techniques for classifying online trade fraud complaints. Combining these initial promising results with the literature studies where [1] machine learning techniques were found to aid crime analysis in general, [2] machine learning techniques were found to hold a high performance when classifying textual data, and [3] the performance of machine learning techniques on skewed data can be improved through data alterations, led to the conclusion that machine learning techniques can indeed be utilized for complaint handling.

9.1.4 SQ4: What machine learning techniques perform best with respect to the automatable steps in the process of handling a complaint?

Answering this sub-question required the combination of all answers to the above sub-questions. In chapter 7, a data analysis was conducted in which each machine learning technique explained by Sebastiani (2002) was evaluated on its ability to predict whether a complaint will be withdrawn or not. Here, the performance of each technique was optimized using a combination of resampling, word normalization, and word weighting. Overall, it was found that through data alterations, the performance of a machine learning technique can be improved up to 13.5pp, with probabilistic classifiers having the highest unimproved performance. By optimizing a classifier it can predict with 80-90% certainty (i.e. precision) that a complaint will not be withdrawn and 30-35% certainty that it will be. Furthermore, it became apparent that the difference in optimized performance between the best classifier (i.e. Logistic regression; 59.4%) and the worst classifier (i.e. K-nearest-neighbor; 55.2%) is only minor with 4.2pp. Four classifiers hold the best optimized classification performance within a range of 1pp, namely the logistic regression, multinomial naive Bayes, multivariate naive Bayes, and association rule classifiers.

9.2 Answering the main research question

The main research question of this thesis was: *"How can the handling of online trade fraud complaints be (partly) automated?"*. Through interviews at both DLOC and LMIO it was found that the process of registering a complaint at DLOC was most relevant to be automated. In specific, by predicting whether a complaint will be withdrawn or not, more important complaints can be dealt with earlier, thereby increasing the processing speed of those complaints at DLOC, while also indirectly improving the process at LMIO.

Based on previous work by Peters (2016a), machine learning techniques were found to be applicable for making this prediction. Literature studies on the use of machine learning techniques for characteristics of the dataset (i.e. criminal, textual, and skewed) revealed the combination of machine learning with data alteration techniques to result in the best classification performance, which was confirmed during the data analysis. The data analysis furthermore revealed that the best optimized machine learning technique to be used for making the prediction is Logistic regression. Using a Logistic regression classifier, a recall of 33.5% and a precision of 29.7% can be attained for the minority class, while for the majority class a recall of 84.0% and a precision of 86.3% can be attained.

As described in chapter 5, the applicability of this research at the Dutch National Police depends upon the classification performance on the “withdrawn” complaints, since these complaints will be stored in a separate bin and will receive a tailored processing approach. Based on the above mentioned results, it is not likely that the Dutch National Police will automatically withdraw a complaint using a Logistic regression classifier. However, since the results for both the minority and majority class exceed the unsampled ratio of minority to majority complaints (16.7%/83.3%), the process of handling an online trade fraud complaint can be partly automated using machine learning techniques. In cooperation with the Dutch National Police, it should be decided whether the classification performance complies with their standards, and how the separate bin will be treated.

10 Future Research

During this research, many interesting opportunities for further research came to light. First of all, it has been opted in this research to use the 100 most occurring features for training a classifier instead of using a less naive feature selection method. In the discussion, the influence of selecting features using bi-normal separation (Forman, 2003) has been evaluated for a multinomial naive Bayes classifier. Resulting from this evaluation was the conclusion that the distinctive features in this dataset do not cover a wide spectrum of the dataset, which does not benefit the performance of a probabilistic classifier. This does not impose, however, that feature selection cannot positively influence other classifiers. Additional research is required to therefore conclude whether feature selection does indeed hold no value when the distinctive features in a dataset are not widespread. Furthermore, it should be explored whether the findings regarding data altering, as performed in this research, hold when using feature selection.

In this research, it has been examined whether a complaint's free-text field can be used to predict whether a complaint will be withdrawn or not. Even though it was concluded that this free-text field can indeed be used to some extent to make this prediction, the performance was restricted. Since a complaint does not solely exist of a free-text field, but contains 59 other attributes, it should be researched if a more accurate classifier can be created using a selection of those attributes, thereby further helping the police in their goal to improve the handling of criminal complaints filed online by Dutch civilians. Based on initial results by Peters (2016a), using a selection of the 59 other attributes could result in an improved classification performance in comparison to using merely the free-text field.

Independent from the rest of this research, it was discussed in chapter 5, that employees at DLOC perceive the answering of complainant's questions received via e-mail as the most time intensive task. To ease this task, a classifier could be trained which evaluates the content of an e-mail to predict the type of question, based on which a reply e-mail is formulated. Unfortunately, no dataset was available at the time of this research. However, when such a dataset can be built, it would be useful for the Dutch Police to have such a classifier support them in their daily activities.

11 References

Abdelhamid, N., Ayesh, A., & Thabtah, F. (2014). Phishing detection based Associative Classification data mining. *Expert Systems with Applications, 41*(13), p. 5948–5959.

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-Based Learning Algorithms. *Machine Learning*, *6*(1), p. 37-66.

Baumgartner, K., Ferrari, S., & Palermo, G. (2008). Constructing Bayesian networks for criminal profiling from limited data. *Knowledge-Based Systems, 21*(7), p. 563–572.

Beleites, C., Neugebauer, U., Bocklitz, T., Krafft, C., & Popp, J. (2013). Sample size planning for classification models. *Analytica Chimica Acta, 760*, p. 25-33.

Bernklau, J., & van der Putte, J. (2015). Jurisprudentie internetoplichting LMIO. Openbaar Ministerie.

Bex, F. (2011). *Arguments, Stories and Criminal Evidence: A Formal Hybrid Theory*. Springer, Dordrecht.

Bex, F., Peters, J., & Testerink, B. (2016). AI for Online Criminal Complaints: From Natural Dialogues to Structured Scenarios*. AI4J–Artificial Intelligence for Justice*, p. 22–29.

Boba, R. L. (2005). *Criminal Analysis and Crime Mapping*. Thousand Oaks, CA: Sage Publications, Inc.

van den Braak, S.W. (2010). Sensemaking software for crime analysis. Doctoral dissertation, Department of Information and Computing Sciences, Utrecht University.

Brause, R., Langsdorf, T., & Hepp, M. (1999). Neural Data Mining for Credit Card Fraud Detection. In *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence.* Chicago, Illinois, USA, p. 103–106.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-Sampling Technique. *Journal of Artificial Intelligence Research*, *16*, p. 321-357.

Chen, H., Chung, W., Xu, J., Wang, G., Qin, Y., & Chau, M. (2004). Crime Data Mining: A General Framework and Some Examples. *Computer, 37*(4), p. 50–56.

Cohen, W. W. (1995). Fast Effective Rule Induction. In *Proceedings of the Twelfth International Conference on Machine Learning.* p. 115-123.

Crombag, H.F.M., van Koppen, P.J., & Wagenaar W.A. (1994). *Dubieuze zaken: De psychologie van strafrechtelijk bewijs* (2e herziene druk). Amsterdam: Contact.

DiMaggio, P. (1997). Culture and cognition. *Annual review of sociology*, p. 263–287.

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive Learning Algorithms and Representations for Text Categorization. In *Proceedings of the seventh international* *conference on Information and knowledge management*. Bethesda, Maryland, USA: ACM, p. 148-155.

Forman, G. (2003). An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research, 3*, p. 1289-1305.

Frank, E., Hall, M., Holmes, G., Kirkby, R., Pfahringer, B., Witten, I.H., & Trigg, L. (2010). Weka-A machine learning workbench for data mining. *Data Mining and Knowledge Discovery Handbook*, p. 1269–1277

Freund, Y., & Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In Vitányi P. (Eds.), *Computational Learning Theory. EuroCOLT 1995. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence)*. p. 23-37. Berlin: Springer.

Freund, Y., & Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. In Saitta, L. (Eds.), *Proceedings of the Thirteenth International Conference on Machine Learning (ICML 1996)*. p. 148-156. San Mateo, CA: Morgan Kaufmann Publishers.

Gao, J., Ding, B., Fan, W., Han, J., & Philip, S. Y. (2008). Classifying Data Streams with Skewed Class Distributions and Concept Drifts. *IEEE Internet Computing, 12*(6), p. 37-49.

Gupta, A., Syed, A., Mohammad, A., & Halgaguge, M.N. (2016). A Comparative Study of Classification Algorithms using Data Mining: Crime and Accidents in Denver City the USA. *International Journal of Advanced Computer Science and Applications, 7*(7), p. 374–381.

Greenwood, P.W., & Petersilia, J. (1975). *The criminal investigation process: Volume I, Summary and policy implications.* Santa Monica. CA: Rand.

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques*. Waltham, MA: Elsevier.

Heuer, R. J. (1999). *Psychology of Intelligence Analysis*. Washington DC: United States Government Printing Office.

Hornik K, Buchta C and Zeileis A (2009). Open-Source Machine Learning: R Meets Weka. *Computational Statistics*, *24*(2), p. 225–232

Inspectie veiligheid en justitie (2015). Aanpak van internetoplichting door de politie: inspectieonderzoek naar een vorm van cybercrime.

Japkowicz, N., & Stephen, S. (2002). The Class Imbalance Problem: A Systematic Study. *Intelligent data analysis, 6*(5), p. 429–449.

Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In Nédellec C. & Rouveirol C. (Eds.), *European Conference on Machine Learning.* p. 137-142. Berlin: Springer.

Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In Mellish C.S. (Eds.), *Proceedings of the 14th International Joint Conference on Articial Intelligence*, p. 1137-1143. San Mateo, CA: Morgan Kaufmann Publishers.

Latukolan, R., & van Ginkel, J. (2016). *Het in kaart brengen van het proces bij initiatie van een melding tot het opstellen van een dossier*. Unpublished internal document.

Le Cessie, S., & van Houwelingen, J. C. (1992). Ridge Estimators in Logistic Regression. *Applied Statistics*, *41*(1), p. 191-201.

Leukfeldt, E.R., Domenie, M.M.L., & Stol, W.Ph. (2010). *Verkenning Cybercrime in Nederland 2009*. Den Haag: Boom Juridische Uitgevers.

Liu, A. Y. C. (2004). *The Effect of Oversampling and Undersampling on Classifying Imbalanced Text Datasets*. Master Thesis. University of Texas, Austin.

Liu, Y., Loh, H. T., & Sun, A. (2009). Imbalanced text classification: A term weighting approach. *Expert systems with Applications, 36*(1), p. 690-701.

John, G. H., & Langley, P. (1995). Estimating Continuous Distributions in Bayesian Classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence,* p. 338-345.

Jurafsky, D., & Martin, J. H. (2014). *Speech and Language Processing.* Upper Saddle River, New Jersey, USA: Prentice Hall.

McCallum, A., & Nigam, K. (1998). A Comparison of Event Models for Naive Bayes Text Classification. In *AAAI-98 Workshop on Learning for Text Categorization*, *752*, p. 41-48.

Minnebo, P. (2004). *Criminaliteitsanalyse Verklaard*. Den Haag, The Netherlands: Elsevier.

Oatley, G. C., & Ewart, B. W. (2003). Crimes analysis software: 'pins in maps', clustering and Bayes net prediction. *Expert Systems with Applications, 25*(4), p. 569–588.

Osborn, D., & Wernicke, S. (2003). *Introduction to Crime Analysis: Basic Resources for Criminal Justice Practice*. Binghampton, NY: Haworth Press.

Pang B., Lee L., & Vaithyanathan S., (2002). Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 79–86.

Peters, J. (2016a). *Classifiers*. Unpublished internal document.

Peters, J. (2016b) *IAC: Tussenrapport*. Unpublished internal document.

Phua, C., Alahakoon, D., & Lee, V. (2004). Minority Report in Fraud Detection: Classification of Skewed Data. *ACM SIGKDD Explorations Newsletter, 6*(1), p. 50–59.

Pirolli, P., & Card, S. K. (2005). The Sensemaking Process and Leverage Points for Analyst Technology as Identified Through Cognitive Task Analysis. In *Proceedings of the 2005 International Conference on Intelligence Analysis*. McLean, Virginia, USA.

Platt, J. (1998). Fast Training of Support Vector Machines using Sequential Minimal Optimization. In Schoelkopf B., Burges C., & Smola A. (Eds.), *Advances in Kernel Methods - Support Vector Learning*. p. 185-208.

de Poot, C. J., Bokhorst, R. J., van Koppen, P. J., & Muller, E. R. (2004). *Rechercheportret: Over Dilemma's in de Opsporing*. Den Haag, The Netherlands: Kluwer.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program, 14*(3), p. 130-137.

Quinlan, J.R. (1993*). C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers.

R Core Team (2015). R: A language and environment for statistical computing. *R Foundation for Statistical Computing*, Vienna, Austria.

Ruiz, M. E., & Srinivasan, P. (1998). Automatic Tekst Categorization Using Neural Networks. In *Proceedings of the 8th ASIS SIG/CR Workshop on Classification Research*. P 59-72.

Schneider, K-M. (2004). On Word Frequency Information and Negative Evidence in Naive Bayes Text Classification. In Vicedo J.L., Martínez-Barco P., Muńoz R., & Saiz Noeda M. (Eds.), *Advances in Natural Language Processing. Lecture Notes in Computer Science, 3230*. p. 474-485. Berlin: Springer.

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys (CSUR), 34*(1), p. 1–47.

Schapire, R. E., & Singer, Y. (2000). BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning, 39*(2–3), p. 135–168.

Sharma, A., & Panigrahi, P. K. (2012). A Review of Financial Accounting Fraud Detection based on Data Mining Techniques. *International Journal of Computer Applications, 39*(1), p. 37–47.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*(4), p. 427-437.

Streefkerk, M. (2015). Zicht op online handelsfraude. *Digitale Criminaliteit*, *11*(5), p. 40–44.

Sun, Y., Kamel, M. S., Wong, A. K., & Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition, 40*(12), p. 3358–3378.

Sun, Y., Wong, A. K., & Kamel, M. S. (2009). Classification of imbalanced data: a review. *International Journal of Pattern Recognition and Artificial Intelligence, 23*(4), p. 687–719.

Tang, Y., Zhang, Y. Q., Chawla, N. V., & Krasser, S. (2009). SVMs modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39*(1), 281-288.

Timmers, N. (2016). *The hybrid theory in practice.* Unpublished internal report.

W. Van Atteveldt, 2014. *Frogr: R client for the frog tagger and parser for Dutch*. R package version 0.100.

Van den Bosch, A., Busser, G.J., Daelemans, W., and Canisius, S. (2007). An efficient memory-based morphosyntactic tagger and parser for Dutch. In F. van Eynde, P. Dirix, I. Schuurman, & V. Vandeghinste (Eds.), *Selected Papers of the 17th Computational Linguistics in the Netherlands Meeting.* p. 99-114.

Wang, F., Wang, Z., Li, Z., & Wen, J. R. (2014). Concept-Based Short Text Classification and Ranking. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*. p. 1069-1078.

Wang, Y., Zhang, Y., & Wang, Y. (2009). Mining Data Streams with Skewed Distribution by Static Classifier Ensemble. In B.-C. Chien & T.-P. Hong (Eds.), *Opportunities and Challenges for Next-Generation Applied Intelligence*. p. 65–71. Berlin: Springer.

Weiss, S. M., Apte, C., Damerau, F. J., Johnson, D. E., Oles, F. J., Goetz, T., & Hampp, T. (1999). Maximizing Text-Mining Performance. *IEEE Intelligent Systems and their applications, 14*(4), p. 63–69.

Wieringa, R. (2010). Design Science Methodology: Principles and Practice. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering – Volume 2*. New York, USA: ACM, p. 493–494.

Witten, I. H., & Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques.* San Fransisco, CA: Morgan Kaufmann.

Wolpert, D. H. (1992). Stacked Generalization. *Neural Networks*, *5*(2), p. 241–259.

Yang, Y. (1999). An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval, 1*(1-2), p. 69-90.

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. Berkeley, California, USA: ACM, p. 42-49.

Yang, Y., & Pedersen, J. O. (1997). A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning.* Nashville, TN, USA: Morgan Kaufmann Publishers, p. 412–420.

Zhang, W., Yoshida, T., & Tang, X. (2011). A comparative study of TF*IDF, LSI and multi-words for text classification. *Expert Systems with Applications, 38*(3), p. 2758-2765.

12 Appendix