Master's Thesis

To Catch a Thief: fraud detection with reliable machine learning

Author:

Rob C. de Wit, BSc

Supervisors:

Dr. Cassio P. de Campos

Dr. Matthieu J.S. Brinkhuis

Iskander van Duuren

Dennis Boose, MSc

Utrecht University

bol.com

Master of Business Informatics
Department of Computing and Information Sciences

June 2019

"*We demand rigidly defined areas of doubt and uncertainty!*"

Douglas Adams

UTRECHT UNIVERSITY

Abstract

Faculty of Science

Department of Computing and Information Sciences

Master of Business Informatics

To Catch a Thief: fraud detection with reliable machine learning

by Rob C. de Wit, BSc

The opaqueness of machine learning (ML) models is a major hurdle for adoption throughout the industry. Actors do not outright trust predictions generated by models. Instead they opt to use business rules-based systems in combination with manual classification, rather than ML models. Reliable machine learning (RML) seeks to mitigate this issue by providing a metric that establishes to which extent the model is certain about its predictions. We investigate the applicability of RML techniques for high-stakes contexts. Specifically, we conduct a case-study in fraud detection at bol.com, the Netherlands' largest online retail platform. Using hand-labelled data from 199,939 orders, we train models that classify those orders as either fraudulent or legitimate. These models are a Naive Bayes classifier (NB), a Credal Sum-Product Network (CSPN), and an XGBoost gradient booster (XGB). All three of these models provide probability as a reliability metrics for their predictions, and the NB and CSPN models provide robustness as a metrics as well. The analysis of robustness in a CSPN with a continous feature is a novelty and extends its application to many new domains. While the overall accuracy of the models does not exceed that of manual classification, we demonstrate how RML can improve upon existing business processes in four manners: 1) providing accurate predictions over a subset of the observations; 2) allowing for a flexible accuracy/cover trade-off; 3) inferring the latent difficulty variable for classifying individual observations; and 4) eliciting features and feature combinations that allow for increased business knowledge. This shows how RML can yield improvements upon existing business processes even when overall predictive accuracy of models is low, validating and building upon existing research in this domain.

Acknowledgements

This project has been my first great endeavour into the world of data science, and especially machine learning. I thank Cassio and Matthieu for their feedback and guidance, because without them this project would have been of far lesser quality.

I am also grateful towards the people at bol.com; particularly the Fraud & Risk and Matching & Collection teams. They provided me with the opportunity to conduct this research and were greatly helpful in my efforts. Most of all I thank them for being so welcoming towards me during my internship.

Lastly, I thank my parents, Paul and Mariella, for their encouragement and empowerment throughout my studies at Utrecht University – and of course far before then.

Rob C. de Wit
June 2019

Contents

List of Figures

List of Tables

List of Abbreviations

AI	**A**rtificial **I**ntelligence
CRISP-DM	**CR**oss **I**ndu**S**try **P**rocess for **D**ata **M**ining
CSPN	**C**redal **S**um-**P**roduct **N**etwork
IML	**I**nterpretable **M**achine **L**earning
ML	**M**achine **L**earning
NB	**N**aive **B**ayes
RML	**R**eliable **M**achine **L**earning
SPN	**S**um-**P**roduct **N**etwork
XGB	**XGB**oost, or e**X**treme **G**radient **Boost**er

Chapter 1

Introduction

In 2017 worldwide potential losses due to fraud in eCommerce amounted to $57.8 billion (Signifyd and PYMNTS, 2017). It is no wonder why online retailers do their utmost to prevent fraud from occurring, with many of them employing state-of-the-art technology to thwart prospective fraudsters. Machine Learning (abbr: *ML*) has been a blossoming field since the 1990s (Langley, 2011), and there have been prototypes for fraud detection through ML since at least 2001 (Lek et al., 2001). Nevertheless, the industry is still in the early days of adopting ML techniques (Deloitte, 2017; Lorica and Nathan, 2018).

One reason for this lagging adoption is that many ML techniques result in a black box: models which are so complex that understanding their reasoning is impossible. This is undesirable and as such a major obstacle in many contexts, such as autonomous vehicles (Koo et al., 2015) and healthcare (Caruana et al., 2015). Acting on the results of black box models can be challenging, because it is unknown how reliable those results are. Reliable Machine Learning (abbr: *RML*) seeks to mitigate this issue by providing a measure of certainty for predictions made by the model.

Problem statement

Fraud detection is another high-stakes example where reliability is an important factor. Basing decisions on unreliable outputs may result in steep losses, both monetarily and in trust from customers. Many organisations instead rely on traditional business rules to detect fraud, but the effectiveness of this approach is highly dependent on the expert's knowledge (Wong et al., 2000).

This research is a case study into the applicability of RML techniques for high-stakes classifications. We specifically investigate how RML can improve existing business processes for fraud detection in eCommerce.

Contributions

The findings produced by this research contribute to the overall state-of-the-art of RML. We apply known techniques in a specific context, and show how business processes can be improved. Specifically, we demonstrate how reliability measures can be used to subset predictions in such a manner that certain accuracy requirements can be met.

1.1 Context

The research described in this thesis was conducted in cooperation with bol.com, the Netherlands' largest online eCommerce platform. They serve 9.8 million customers in the Netherlands and Belgium and had a €2 billion revenue in 2018. The organisation is still in the exploratory phase of using ML to further its business needs and have asked us to research the possibilities for applying ML in order to classify fraudulent orders from customers. While bol.com started out as a webshop, its focus is shifting to becoming an online e-commerce platform. Because of this, the organisation is looking for solutions that can be provided as a service to partners as well.[1]

Fraud & Risk management

The Fraud & Risk team is part of the Platforms Operations division within bol.com. One of their responsibilities is to review the orders from customers which got flagged by a system consisting of business rules. Of all orders, 0.2%* are reviewed with the reviewing expert deciding on a case-by-case basis whether to accept the order. Orders that are classified as fraudulent are cancelled, with the customer being notified of the decision. Because bol.com wants to minimise fulfillment time, these reviews are a 24/7 process, accounting for 30%* of the team's workload.

1.2 Research approach

We view this research through the lens of the *design science* paradigm, which Wieringa (2014) describes as "the design and investigation of artifacts in context". Within design science the problem is an improvement to be made within a context, which is achieved through the design of an artifact. This artifact is constructed through application of existing knowledge and newly gained knowledge from the investigation itself.

As part of this research approach we specify the Research Goal and associated research questions throughout the following section.

1.2.1 Research goal

This research specifically looks at what Wieringa (2014) characterises as an *Artefact Design Goal*: through creation of an artifact, we strive to improve a context. In accordance with the goal definition framework we specify the following design problem:

"This research aims to improve fraud detection by designing a reliable machine learning model that classifies fraudulent orders with acceptable accuracy so that bol.com can increase working efficiency."

Specifically the accuracy should be on par with performance from the experts who currently manually review possibly fraudulent orders.

[1] In order to maintain confidentiality of bol.com's business metrics, we cannot always provide precise numbers. Instead, this thesis uses rough approximations for these business metrics in order to sketch a general idea of the context. Numbers where this is the case are indicated with an asterisk (*).

Given the lack of experience with ML at bol.com it is unrealistic to expect outright replacement of manual assessment with classification by a model. Nonetheless, this would be an exciting prospect if the solution proves to be accurate enough and as such a possible next step. Hence, a secondary objective of this project is to gain experience with (R)ML, thus building trust in its capacities and creating a basis for future ventures.

1.2.2 Research questions

In order to address the specified research goal we want to investigate how RML can enhance existing business processes for detection of fraudulent orders. More generally, we can translate this into the following research question around which this thesis is structured:

"How can reliable machine learning be applied to improve upon existing business processes for high-stakes classifications?"

This research question can be decomposed into three distinct sub-questions that contribute to answering this main research question:

1. Why are machine learning models not being widely used for high-stakes classifications?
2. What metrics can be used to determine the reliability of classifications made by machine learning models?
3. Which machine learning approaches can be applied for high-stakes classifications?

1.3 Thesis outline

The rest of this thesis is divided into five chapters. Firstly, related works are discussed to provide background to ML for fraud detection, as well as the domains of interpretable and reliable ML. Chapter 3 discusses the methods we apply in order to attain our research goal. The application thereof is discussed in chapters 4 and 5, which together form the most significant part of this thesis. Lastly, we provide a summary, conclusions, and limitations in chapter 6, as well as future research opportunities.

Chapter 2

Related works

"Standing on the shoulders of giants" is a well-known metaphor within science. It illustrates how we can see new things by building upon previous discoveries. This thesis is no exception to this wisdom, and likewise builds upon discoveries from many different researchers. To show the existing foundations for this project, we review relevant topics within the scientific literature.

Because of the breadth of the research domain, this chapter is by no means an exhaustive review. Rather, it should be considered a glimpse at the domain this research takes place in. Throughout the following chapter we first provide a brief introduction to the concept of ML. We then evaluate the state of the art for fraud detection with ML, focussing on eCommerce fraud in particular. Furthermore we provide elaboration on the concept of RML and discuss relevant research conducted in this domain.

2.1 Machine learning

ML is a sub-domain of Artificial Intelligence (abbr: *AI*), the latter of which seeks to create and analyse agents that intelligently solve tasks when provided with information or stimuli (Russell and Norvig, 2016). The term *Machine Learning* was first coined by Samuel (1959), who stated that ML "gives computers the ability to learn without being explicitly programmed". More recently it has been interpreted as a field of study that focuses on computational approaches for learning models from data to best perform certain tasks (Mohri, Talwalkar, and Rostamizadeh, 2012). In other words, rather than provide a machine with a predefined algorithm by which it should perform a task, ML equips machines with tools to develop their own approaches. This approach, or *model*, takes inputs and generates outputs as solutions to the defined tasks.

ML techniques can broadly be divided into three categories: supervised, unsupervised, and reinforcement learning. Witten et al. (2016) argue that philosophically speaking, supervised learning would be better characterised as "trained learning": the machine we want to learn is provided with scaffolding as to know what it should be learning. In this approach we provide the machine with a dataset and a list of known outcomes, with the intention of having the machine learn to predict the right outcomes based on the data. In unsupervised learning, on the other hand, neither the goal nor the outcomes are known beforehand. The machine is truly learning on its own accord, and its performance cannot be measured as objectively as in supervised learning (Maini and Sabri, 2017). Reinforcement learning is unique in that it can affect the environment in which the machine is learning. The outcomes are not known beforehand, but the

machine learns the effects of its actions and determines what it needs to do in order to achieve a desired goal.

Detection of fraudulent orders can be considered a classification problem, which is a type of supervised learning. ML techniques geared towards classification take features and labels as input, where the labels are the different classes that the resulting model is to predict. Existing observations are analysed to determine which features contribute to certain classifications. As such, feature weight is often an important aspect of classification models.

2.2 Fraud detection and machine learning

As mentioned in chapter 1 there have been ML prototypes for fraud detection in eCommerce since the beginning of this century. Lek et al. (2001) proposed a data mining method that analysed fraud patterns to generate rules for identifying fraud. Adoption of this approach was not described in scientific literature, however.

Most ML for fraud detection in eCommerce has focused on detection of credit card fraud, although relatively few articles detailing real-world applications have been published (Zareapoor and Shamsolmoali, 2015). Stolfo et al. (1997) and Chan et al. (1999) describe initial experiments in this context. They argue that true positives and false positives are better metrics than overall accuracy, since fraudulent cases are a slim minority of all cases. More generally the problem posed by skewed distributions was evaluated by Kubat, Matwin, et al. (1997), who identified multiple possible solutions. These solutions include weighing instances (Pazzani et al., 1994) and including different misclassification costs for positive and negative instances (Gordon and Perlis, 1989).

Outside of eCommerce, Fawcett and Provost (1996) described automated methods for detecting fraudulent cellphone usage. They found that the available ML methods were less accurate than hand-coded business rules, but the latter took several months to implement whereas ML models only took a few hours. Moreover, they demonstrated the adaptability of ML methods, which enhances the increased effectiveness when compared to business rules.

The context of our current research shares aspects described by these authors. The classes of orders (legitimate or fraudulent) are unevenly distributed, although this skewness is less pronounced in the subset that is manually reviewed than in the overall population. Moreover, the misclassification costs in our context are uneven for the two classes: shipping a fraudulent order is far more costly than not shipping a legitimate order. Another shared aspect is the existing solution in place at bol.com for detecting fraudulent orders: a system of hand-coded business rules. While the accuracy of this system is good, it suffers from the same downsides described in literature.

2.3 Interpretable machine learning

The works above cover the applicability of traditional ML techniques (i.e.: black-box methods) to the domain of fraud detection, but they do not account for the limitations these methods contribute. As illustrated by Koo et al. (2015) and Caruana et al. (2015) the lack of trust in ML models has been a major hurdle for adoption.

Conversely, the opaqueness of black-box models can be considered an asset in certain contexts. Burrell (2016) argues that within domains such as network security or — indeed— fraud detection transparency is often avoided because it provides adversaries valuable insights in the weaknesses of a system: if you do not understand how a system works, neither can your adversaries. Nevertheless, this obscurity is generally considered a downside of ML rather than an asset.

Interpretable Machine Learning (abbr: *IML*) has evolved as field of research that seeks to mitigate this issue. IML seeks to provide interpretability of how predictions came to be. This allows for human understanding as well as explanation of decisions, which may be a requirement in accordance with laws and regulations such as the General Data Protection Regulation (Council of the European Union, 2016). An important distinction is to be made between external interpretability and internal interpretability, or that of the learning algorithm and that of the model itself (Rüping et al., 2006). External interpretability is often (although not always) implicit: a learning algorithm was devised by a human, so a human must be able to understand it as well. This does not automatically result in internal interpretability, however. With an algorithm one understands, it is still possible to create a model that is beyond comprehension.

From a societal point of view there is merit to IML as well. Pasquale (2015) calls for auditors who have access to algorithms to ensure organisations do not use discriminating classifiers.[1] This is, however, impossible when algorithms are designed in such a way that their inner workings cannot be interpreted by humans.

Lack of trust in the outputs of a ML model has been described as a hurdle for adoption at bol.com as well. This showcases the need for ML models that provide interpretability of their predictions within the industry. Bol.com especially wants to be able to interpret how certain they can be that models provide correct outcomes, and this is where reliable machine learning comes into play.

2.4 Reliable machine learning

RML can be considered a sub-domain of ML that specifically makes one aspect of ML models interpretable: it seeks to provide a metric by which we can assess the extent to which predictions are reliable, thus allowing for interpretation beyond "the model thinks this instance belongs to class A". The concept has been described by many authors, some of whom use terminology such as *credibility* and *statistical confidence* (Saunders, Alexander Gammerman, and Volodya Vovk, 1999; Hamilton, Shan, and Ziarko, 1997) as an alternative to *reliability*. Vladimir Vovk, Alex Gammerman, and Shafer (2005) call predictions which incorporate an indication of their accuracy and reliability *hedged predictions*.

A major benefit of reviewing reliability metrics, besides the increased trustworthiness of the model's predictions, is that it allows for subsets of predictions based on instances' scores. One could, for example, accept the predictions with a high reliability score and manually review instances with low reliability. In other words, it becomes possible to control the number of erroneous predictions by selecting a certain reliability metric (Alexander Gammerman and Vladimir Vovk, 2007).

[1] From a technical point of view all classifiers are discriminating, but in this context we colloquially refer to discrimination in the legal sense — or rather the illegal sense.

This concept of reliability was explored by Balasubramanian, Ho, and Vladimir Vovk (2014) and Bosnić and Kononenko (2009), the latter of whom note that it is a useful indicator in contexts where acting upon individual predictions has significant consequences. Examples of these contexts include medical diagnoses and financial decisions. Bosnić and Kononenko recognise the difference between reliability metrics that are model-dependent and -independent. The first type of metric is built into the model and generally probabilistic, whereas the second type treats the model as a black box and analyses how variations in instances affect classification.

Probabilistic model-dependent metrics —the metrics themselves are usually called *probability*— have been described since the 1990s (Cestnik et al., 1990). Kukar and Kononenko (2002) define reliability as "an estimated probability that the classification is the correct one". If an instance has a 0.18 probability of belonging to class A, for example, the prediction of the instance being in class B has a reliability of 0.72.

Robustness is an example of a model-independent reliability metric, which has been shown to provide a better estimate of reliability in certain contexts (Mauá et al., 2017; Conaty, Del Rincon, and De Campos, 2018). Robustness can be considered the extent to which variations can occur without affecting the predicted class. If these variations are large, one can assume an instance to belong firmly to one class, or to be *robust*. This concept is not dissimilar from what was proposed by Weigend and Nix (1994), who showed that predicted variance estimates can serve as a reliability metric.

We consider the fraud domain to be a high-stakes domain where acting upon individual predictions has a significant monetary impact. Probability and robustness are two metrics that can identify in which instances a model's predictions are reliable. We will apply these concepts and analyse how they impact classification of fraudulent orders, and therewith the business processes of bol.com.

Chapter 3

Methods

As described in chapter 1 we view this research through the lens of design science as described by Wieringa (2014). This paradigm iterates over designing and investigating, and Wieringa provides the Engineering Cycle as a structure for doing so. This structure consists of four phases:

1. Problem Investigation: investigate the context, including the causes of the problem, goals, and stakeholders.
2. Treatment Design: specify the requirements of the envisioned solution and design the solution, either through application of available treatments or through creation of a new one.
3. Treatment Validation: determine if the designed treatment solves the problem.
4. Treatment Implementation: integrate the solution and solve the problem.

While we do want to have a framework that defines a structured approach towards solving our design problem, the Engineering Cycle is too generic for our needs. It is not specifically tailored towards the application of ML, but instead sterns from disciplines such as industrial product engineering (Jones, 1980) and systems engineering (Hall, 1962). An approach that better suits the context of this research is the *Cross-Industry Standard Process for Data Mining* (abbr: *CRISP-DM*) (Chapman et al., 2000). CRISP-DM is the most widely-used approach amongst data scientists (Brown, 2015b), and we have previous experiences in following the process. This makes the method both a convenient and logical choice for this research. The following section provides a brief description of CRISP-DM. After that we discuss modifications we make to the method to better suit our needs, detailing how it will be followed in the remaining chapters of this thesis.

3.1 CRISP-DM

CRISP-DM prescribes an iterative process consisting of six phases. The phases, which are visualised in figure 3.1, are as follows:

1. Business Understanding: gain knowledge of the context, the problem we are trying to solve, and how it impacts the organisation.
2. Data Understanding: evaluate the available data and see what we will have to work with.

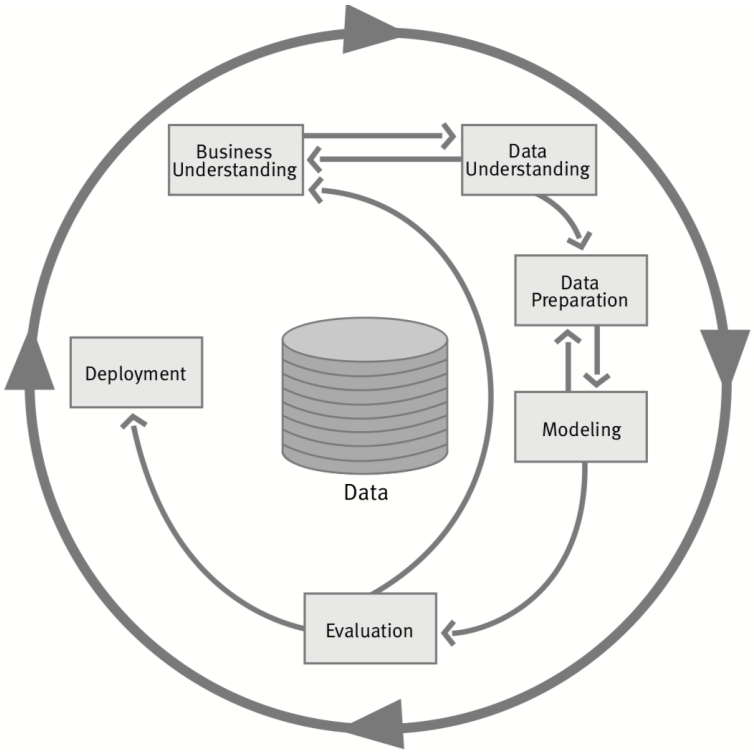


FIGURE 3.1: A visualisation of the CRISP-DM process. Pictured are its six phases (the boxes) and the most prominent interdependencies between those phases (the arrows) (Chapman et al., 2000).

3. Data Preparation: take the "raw" data and transform it into the form required for analysis.
4. Modelling: use techniques to model the data in such a way that it contributes to solving the problem.
5. Evaluation: determine whether the models produced in the previous phase solve the problem.
6. Deployment: integrate the solution into the organisation.

It should be noted that the six phases are neither strictly sequential nor separated, and that advances in one phase can lead to advances in another. For example: transforming time stamps (data preparation) in the data can lead to an understanding of the time the data comes from (data understanding). Written descriptions of the process —such as this thesis—, tend to forego these nuances in favour of a well-framed sequence of events. We ask the reader to keep this in mind throughout the rest of this work.

The phases in CRISP-DM can be broken down into generic tasks, which in turn can be divided into specific tasks. An example of a generic task for Data Understanding is *Collect Initial Data*, which has the *Initial Data Collection Report* as an output. Generic tasks can be further divided into specific tasks, or activities. The relevance of these activities is dependent on the context of the project: "[CRISP-DM] gives detailed tips and hints for each phase and each task within a phase, and depicts how to carry out a data mining project". Throughout chapters 4 and 5 we follow the structure of the generic tasks to describe the specific tasks, i.e. the activities we have concretely

conducted. An overview of the generic tasks and their associated outputs is provided in appendix A.

The observant reader may have noticed that the phases in CRISP-DM are not entirely different from those in the Engineering Cycle. To illustrate, *Business Understanding* in CRISP-DM encompasses tasks that yield the same output as *Problem Investigation* in the engineering cycle, such as the needs and constraints of the organisation. Likewise, *Evaluation* shares its core concept with *Treatment Validation*: determining if the proposed solution adequately satisfies the design goal.

3.2 Adaption

Having selected the base method for this thesis, we will now describe where we deviate from the activities prescribed by CRISP-DM. Specifically, because of the nature of this thesis project, we will not follow the Business Understanding phase. As Brown (2015a) establishes, the most important goal of the Business Understanding phase is to make sure management agrees with the need for data analysis and buys into it. In our case, however, the client has already assented to the need for and goals of this research. More importantly, the problem we aim to solve has already been established in the previous two sections of this thesis. This diminishes the risks the Business Understanding phase is intended to mitigate.

Secondly, we will not describe the evaluation phase as a separate section. The purpose of this phase is to evaluate the project, draw conclusions, and determine what steps are to be taken next. In a thesis such as this one, this shows parallels with the discussion chapter. Hence, we will incorporate these aspects in our final chapter.

Lastly, the Deployment phase will not be conducted for it is outside of the scope of this research. We sadly do not possess the knowledge of bol.com's infrastructure, nor the resources, to facilitate this step. We will, however, take implementation into consideration when selecting data sources and the modelling techniques.

Chapter 4

Data and set-up

This chapter details the first half of the data mining process we described in chapter 3. Each section relates to one of the six phases within the CRISP-DM cycle and describes its outputs.

4.1 Data understanding

The goal of the Data Understanding phase is to understand what data we need to meet the objectives. Specifically, we need to collect data that pertains to orders and includes both features and labels.

4.1.1 Collect initial data

The data for the project were provided to the researchers by bol.com. We agreed that the available data should be identical to or closely resemble the data used with bol.com's production environment. This way a resulting model could be incorporated in the business processes without additional data sources having to be made available. Furthermore, for the purpose of labelling, data should be available that shows the legitimacy of orders after the fact.

This resulted in four related datasets pertaining to four distinct business concepts:

1. Order data: generic data pertaining to orders and the customers who placed those orders. These includes data such as total price, shipping address, date of birth, and gender.
2. Fraud function data: data that are the result of function executions in the existing business-rules. These functions return a value based on an input and determine, for example, whether an IP address is on a blacklist or whether the customer has a record of not fulfilling their payments.
3. Payment status data: data that show whether payments for an order have been fulfilled, gone to debt collection, or been written off. This can be used to identify true negatives and false negatives.
4. Customer support data: data that show whether a customer contacted customer support after their order had been cancelled. This would indicate that the customer was placing a legitimate order, thus indicating false positives.

The observations in each of these datasets were combined through their shared identifier; `ORDER_ID`. Initially, for the purposes of describing and preparing the data, the

number of orders was limited to 36,707. A model can be more accurate, however, when the size of its training set increases. Hence, this number was later increased to 260,189, which encompassed all orders that were cancelled (either automatically or manually) or approved by an expert. Later this number was brought down to 199,939, having left out orders that were not manually reviewed.

The datasets were provided to us as CSV sheets and loaded into RStudio (v1.2.1335) for analysis.

4.1.2 Describe data

Below we will provide descriptions of what data are contained in the acquired datasets. A tabular overview can be found in table 4.1.

Order data

The order data consisted of data relating to both orders (e.g.: `Shipping_location`, `Payment_method`, `Total_price`) and the customers that placed those orders (`City`, `Gender`, `Date_of_Birth`, `Account_type`, and `Date_of_first_order`). The dataset contained 225,195 records, but one order can have multiple records if it contains more than one item. In total, there were 139,729 unique orders in this dataset.

Fraud function data

The fraud function data contained 14,115,810 records, each representing a function executed by the business rules. Exact duplicates (from when a rule was executed by multiple rules) had already been eliminated at query level. The dataset did contain many instances of duplicate executions with differing results, however. This could be the case if a parameter in the function execution was different (e.g.: `CheckIfHigher(50, amount)` and `CheckIfHigher(100, amount)` can return True and False respectively when amount equals 75). The dataset did not contain these parameters.

Payment data

The payment status data consisted of 81,282 records, each of which represented an invoice. Not every order has an invoice, because some orders are cancelled by an expert. Conversely, an order can have more than one invoice if it consists of orders from different suppliers. Aside from `ORDER_ID` the dataset includes `PAYMENT_METHOD` (nominal), `TOTAL_AMOUNT` (ratio), `OPEN_AMOUNT` (ratio), and three different versions of `WRITE_OFF` (ratio; categorised depending on the reason for the write-off).

Customer support data

The customer support data contained 4,471 records, which included `ORDER_ID` and `LOGPATH` at different levels. The latter of these indicates the reason for a customer contacting support. In short, this showed whether a customer had contacted bol.com's

support about one of the orders in the datasets described above, and why they had done so.

Dataset	Features	Records	Description
Orders	15	225,195	General descriptive data of the orders and personal information of the customers
Function executions	6	14,115,810	Functions triggered by the fraud module and their outputs per order
Payments	6	81,282	Invoices and their payment status, including write-offs and debt collection
Customer support	5	4,471	Cases relating to orders

TABLE 4.1: Descriptive overview of the collected datasets: these four datasets contain various aspects of orders that were analysed by bol.com's fraud module from October 2018 to January 2019

4.1.3 Explore data

Off-the-bat the datasets yielded few insights that could help reach our research objectives. Further processing and data combination was needed to allow for application of ML techniques. Specifically, the datasets and their records needed to be combined so that each record pertained to one ordered item. We utilised this step to determine what measures could be taken without significant loss of information.

Initial exploration of the fraud function dataset showed why it had many more records than the number of orders included in the dataset. As we discussed, these functions could return varying results depending on the parameters used in the function call.

The order dataset revealed that some of the included orders had not been manually reviewed by an expert, but were cancelled for other reasons. These cancellations were attributed to either the buyer or seller.

No surprises were found in the payment and customer support datasets, which were both to be used for labelling. If anything, these included more data than necessary for labelling of the orders, such as the precise logpath for customer support cases.

4.1.4 Verify data quality

The quality of the data was determined to be relatively high, with very few oddities. Throughout the datasets there were only 23 NA values, each of which belonged to empty rows.

The only dataset that showed strange values was the order data, where 1.01% of all orders had scrambled personal details. This was due to the customers deleting their

account between the time of ordering and our analysis.[1] Data strictly pertaining to the order itself was preserved, however.

4.2 Data preparation

Data was provided to us in CSV files by bol.com, which we then imported into R. The code written to import and process the data can be found in appendix B. The columns were in some cases renamed to be more convenient for processing, and their contents were converted to appropriate data types. Two columns without any variation were removed. A few conversions were applied as well, namely:

- Converted *true* and *false* strings to logical.
- Converted Y and N to logical.
- Converted factor columns with two levels to logical.

4.2.1 Select data

As discussed in 4.1.3, the function execution dataset contained functions with different outputs depending on the provided parameters. Because we did not have access to the parameters, we decided to remove these observations. This decreased the number of functions in this dataset from 88 to 55.

In consultation with bol.com, personal data such as gender and date of birth was removed from the dataset. While these features are potentially useful, bol.com had reservations about using them in classifications from ML models for ethical and legal reasons. Moreover, not all of these data were available in bol.com's cloud infrastructure at the time of writing. This went against our aim of only using data that would be available in a production environment, so we decided to leave out these features for this project.

Because of restrictions on available computing power, features with little variation (where the minority class accounted for less than 0.01 of the observations) were removed from the datasets. While we would have preferred to not sacrifice these data, exploratory modelling showed that the features had little impact on accuracy of the models.

Lastly, there were some features that occurred across multiple datasets, such as `orderTotal`. These duplications were removed.

4.2.2 Clean data

While some data clean-up did take place, all of this pertained to personal data features that were removed for the reasons elaborated upon. Nevertheless, we will briefly highlight these steps for they could provide insight in situations where one wants to retain those features:

[1] Although we do not possess data to verify this, it is likely that the percentage of "forgotten" accounts is higher in this sample than in the total number of orders. According to business experts, many customers delete their account upon learning it has been compromised, rather than changing its log-in credentials.

- The data contained some records (approximately 1.01%) pertaining to "forgotten" accounts. The owners of these accounts requested to have their personal data removed from bol.com's systems, to which bol.com complied by overwriting their personal data with random strings. These randomised values were converted to missing values.
- Because of the number of levels in certain features (e.g. `PickupPoint` and `City`), these values were converted to the ratio of their occurrences.

4.2.3 Construct data

A number of additional features were created from the existing data to facilitate labelling:

`Total loss` was created by combining different types of write-offs and the open amount for each order. This combined total was used throughout the rest of the project, rather than the more granular features. After all, the being able to distinguish between these various costs is irrelevant for classification of orders: each order that is not going to be paid for, should ideally not be shipped.

We created a feature to determine whether an account had been taken over by an adversary. We did so based on the existing features `IpDiffersFromPreviousOrders` and `AddressDiffersFromPreviousOrders`. The logpaths from the customer support data were also taken into account for this feature. This was done to allow for better labelling: if an account has been taken over, then the customer contacting support does not automatically indicate erroneous cancellation of the order as would normally be the case.

Despite having many features throughout the four datasets, labels were not readily available. Labels were based on the verdict provided by business experts and constructed using the following logic:

- **True Positive:** an order was cancelled and the customer either did not contact support, or had their account taken over and did contact support.
- **True negative:** an order was approved and subsequently paid for.
- **False positive:** an order was cancelled, but the customer contacted support.
- **False negative:** an order was approved but not paid for.

It should be noted that these labels may not be completely accurate. False positives are especially difficult to detect, because we cannot know whether all customers contact support after their order is cancelled. Some might simply order from a competitor or forego their order. Therefore the data can contain orders that are labelled as true positive while in reality being false positives. Construction of this data allowed for analysis of expert performance, which turned out to be 93.25%.[2]

These assessments of correctness were converted into class labels, the ratio's of which are provided in table 4.2. As shown, the positive cases are a minority, although not one as slim as in the overall population of orders.

[2]Because of the difficulties in labelling positive observations described above, the real accuracy is expected to be somewhat lower. How much "somewhat" is, we sadly cannot know.

	Legitimate	Fraudulent
Total observations	157311	42628
Ratio	0.79	0.21

TABLE 4.2: Frequency of classes within the dataset: the fraudulent cases are a minority within the orders that are manually reviewed by experts.

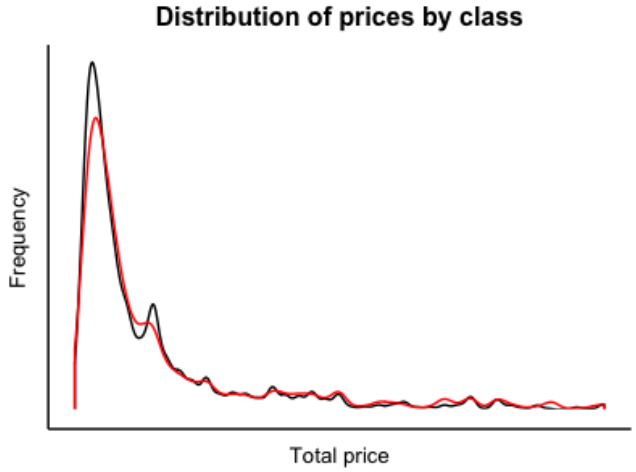


FIGURE 4.1: Distributions of price within the dataset: distributions for fraudulent observations (red) are comparable to those of legitimate observations (black).

As visualised in figure 4.1, we found that the one continuous feature in the dataset (i.e. `Total_Price`) had similar distributions for both the legitimate and fraudulent classes.[3]

4.2.4 Integrate data

We integrated the datasets by `OrderID`: one observation for each manually reviewed ordered item that contained the function execution results, customer support data, payment data, additionally constructed columns, and a label determining fraud. This resulted in a matrix with $199,939$ rows and 55 columns.

4.2.5 Format data

The data was one-hot encoded to ensure compatibility, since many implementations of ML techniques are only able to process 1s and 0s. This converted features with more than 2 levels into separate columns.

The resulting dataset after all of these tasks contained 67 columns and $199,939$ rows. All columns were logical except for `Total_Price`, which was numerical. A full overview of the features (pseudonymised) and their distributions is provided in appendix C.

[3] In order to maintain confidentiality with bol.com, the numbers in this figure are redacted.

Chapter 5

Execution and results

The following chapter describes the processing of the dataset into fully fledged models, as well as an analysis of these models' results. This corresponds with phases 4 and 5 of CRISP-DM: modelling and evaluation.

5.1 Modelling

Now that we had availability over well-structured data, we could start what many data scientists consider the exiting part: modelling! Given the iterative nature of tweaking models and improving the performance, we cannot truly describe the process in a sequential manner. Therefore, the following section can be considered to be a description of the modelling choices we eventually decided upon and the motivations that led to these choices.

5.1.1 Select modelling techniques

We decided to select three distinct ML techniques as candidates for creating an RML model. This allowed for a three-way comparison of their performance, resulting in an analysis on how different reliability metrics affect classifications in different circumstances. Additionally, a three-way comparison could show the generalisability of the concept of RML in high-stakes contexts.

We wanted to select a set of techniques that included both a well-studied approach and one that is considered state of the art in the industry. In addition to this, we also wanted to analyse a relatively new technique that showed promise but has not been studied extensively yet within our context.

Aside from diversity being a criterion, the ML techniques also had to work within the limited technical scope that was available to us. This meant that it was necessary for models to allow for local training and testing, while also providing opportunities for operationalisation on a larger scale in the future. As such, ML techniques with excessive processing requirements were ruled out.

After careful deliberation, we decided upon the following three candidates, each of which adhered to the requirements outlined above:

- Naive Bayes
- Credal Sum-Product Networks
- XGBoost

Each of these ML techniques provides probability as a model-dependent reliability metric. The latter two also provide robustness of classifications as a metric. In the following subsections we will briefly describe the background and workings of each of the candidates, as well as our motivations for selecting these models specifically.

Naive Bayes

A Naive Bayes (abbr: *NB*) classifier is a classifier based on Bayes' theorem. It was proposed by Maron (1961) and uses probabilities of independent features belonging to certain classes to determine to which class observations most likely belong. While rudimentary in nature, training a NB classifier is relatively fast and its results often show high accuracy (Stecanella, 2017). A downside of NB is its reliance on the independence of features, which is not necessarily the case within this context.

Since its introduction it has been used extensively in classification tasks and become a de facto baseline technique (Wang and Manning, 2012), making NB a logical choice for a classifier in a three-way comparison. Comparing the other classifiers against a NB classifier showcases their performance relative to technique that is widely applied throughout the industry.

Credal Sum-Product Network

Credal Sum-Product Networks (abbr: *CSPN*) (Mauá et al., 2017) are a variation on Sum-Product Networks (abbr: *SPN*) (Poon and Domingos, 2011). SPNs were designed as a deep architecture, posing an alternative to graphical models such as NB. SPN classifiers are well suited for producing inferences, meaning they are tractable over trees with a large width. As such, no approximation techniques for the probabilities of features have to be used. An SPN is comprised of nodes that perform weighted sums or multiplications to predict an instance's class. Whereas NB classifiers rely on independence of features, SPNs can process combined features through those sums and multiplications. Gens and Pedro (2013) show that SPNs are typically superior to graphical models in inference speed and accuracy.

CSPNs provide an imprecise extension to SPNs, thus facilitating the representation of incomplete knowledge. This mitigates SPNs' susceptibility to unreliability and overconfidence. CSPNs achieve this by allowing the weights of sum nodes to vary within a certain range. Through this process it can be established to what degree these weights can vary without affecting the outcome class. CSPNs provide this extent as a robustness value, which can be used to assess the reliability of predictions as described in section 2.4.

CSPNs have been a recent subject of study at Utrecht University, providing a considerable in-house knowledge base on this ML technique. Accessibility to this knowledge and being able to contribute to on-going research was a major motivation for selecting this technique. While CSPNs show promise, little research has been conducted into their real-world applications. Using this method on a dataset such as ours, which includes a continuous numerical feature, would be a novel contribution to the research domain.

XGBoost

XGBoost (abbr: *XGB*) is an implementation of tree boosting that is widely used within the industry and outperforms similar implementations (Chen and Guestrin, 2016). It is based on the principle of gradient boosting (Friedman, 2001), which in turn is an extension of adaptive boosting, or AdaBoost (Freund, Schapire, and Abe, 1999).

Gradient boosting techniques combine subsequently trained boosters that aim to achieve marginally better results than the previous iteration. Through this approach weak learners are assembled into one strong learner. Although the approach produces accurate results, it has the downside of being greedy. XGB is an implementation that specifically optimises for performance. It has become the leading model in Kaggle competitions, where its prevalence is attributed to its ease of implementation and state of the art performance and speed (Becker, 2018; Pathak, 2018; Brownlee, 2016).

Perhaps as a result of its widespread use, XGB is one of the techniques readily available within the Google Cloud platform, which is part of the existing IT infrastructure at bol.com. This would allow for relatively easy operationalisation in a production environment, motivating our choice for this candidate. Additionally, minor experiments using XGB have been conducted at bol.com, providing some in-house knowledge.

5.1.2 Generate test design

The dataset was split into two separate sets for training and testing. We decided upon a division of 0.80 to 0.20, because it maximises learning while still retaining a sufficiently large test set. The division was stratified on the labels of the observations (i.e.: fraudulent or legitimate), ensuring comparable distributions between the two sets.

Aside from the "raw" predictions for the observations in the test set (i.e.: the binary classifications), the reliability metrics were also recorded. As discussed before, all three models provided probabilities for their predictions. Additionally, the NB and CSPN classifiers recorded the robustness of those predictions.

All three models were trained on and tested against the exact same datasets, although additional feature selection was applied for the NB and CSPN classifiers to reduce processing time. This was done based on the variance within features, with only the logical features where the minority class comprised at least 0.01 being selected.

5.1.3 Build model

Building the models was an iterative process, in which we tweaked hyper-parameters and evaluated their effects. The code for building the models can be found in appendices D and E.

Whereas the XGB classifier was relatively quick to train (approximately 30 minutes), the NB and CSPN classifiers took multiple hours and were generally trained overnight. Because the implementation (not published) for the NB and CPSN classifier could not process missing values, we replaced these values with the median of those features. XGB classifiers are able to recognise extreme outliers, so for this model we replaced the missing values with the maximum negative integer value (-2147483648).

NB	CSPN	XGB
thr = 0.01 height = 2 samples = 5	thr = 0.01 height = 1,000,000 samples = 5	nround = 1000 eta = 0.01 max.depth = 20 min_child_weight = 10 scale_pos_weight = $\#false/\#true$ lambda = 2

TABLE 5.1: Overview of hyperparameters: these settings were used in the training of the three classifiers.

The hyper-parameter settings for the NB and CSPN models were determined in cooperation with the creators of the implementation (not published), prioritising training time. The settings for the XBG classifier were determined through a grid-search as inspired by Revert (2018), optimising for overall prediction accuracy. The resulting settings for all three classifiers can be found in table 5.1. It should be noted that the parameters are not comparable between the different classifiers.

5.2 Assessment

We assess the performance of the models based on two underlying metrics, the first of which is their predictive accuracy. Secondly, we use *Cover* extensively throughout the following sections. We define this metric as the proportion of the dataset's observations for which the model assigns a prediction. A cover of 0.2, for example, equates to the observations with a reliability measure equal to or above the 80th percentile. This allows us to analyse a model's accuracy over the n predictions with the highest reliability score.

The evaluation was performed in two phases. First, we extensively analysed the performance of the robustness metric for the CSPN classifier over a subset of the dataset. These findings were published in Rob C. De Wit et al. "Robustness in Sum-Product Networks with Continuous and Categorical Data". In: *Proceedings of the 2019 International Symposium on Imprecise Probabilities*. 2019, which is also included in appendix F.

We thereafter analysed the performance of all three classifiers and their associated reliability metrics against one another over the full dataset. The following subsections discuss these respective analyses.

5.2.1 Robustness with continuous data

As described in section 5.1.1, CSPNs have been a recent subject of study at Utrecht University. One of the aspects that has been studied is the extension of CSPNs to process continuous features. The dataset from bol.com provided an opportunity to conduct a case study on a real-world application, validating the effectiveness of robustness as a reliability metric.

Because we did not possess the entire dataset at the time of publishing these results, we performed the analysis on a subset of the dataset. This subset consisted of of all orders placed in October of 2018. This was the same subset as the original dataset of

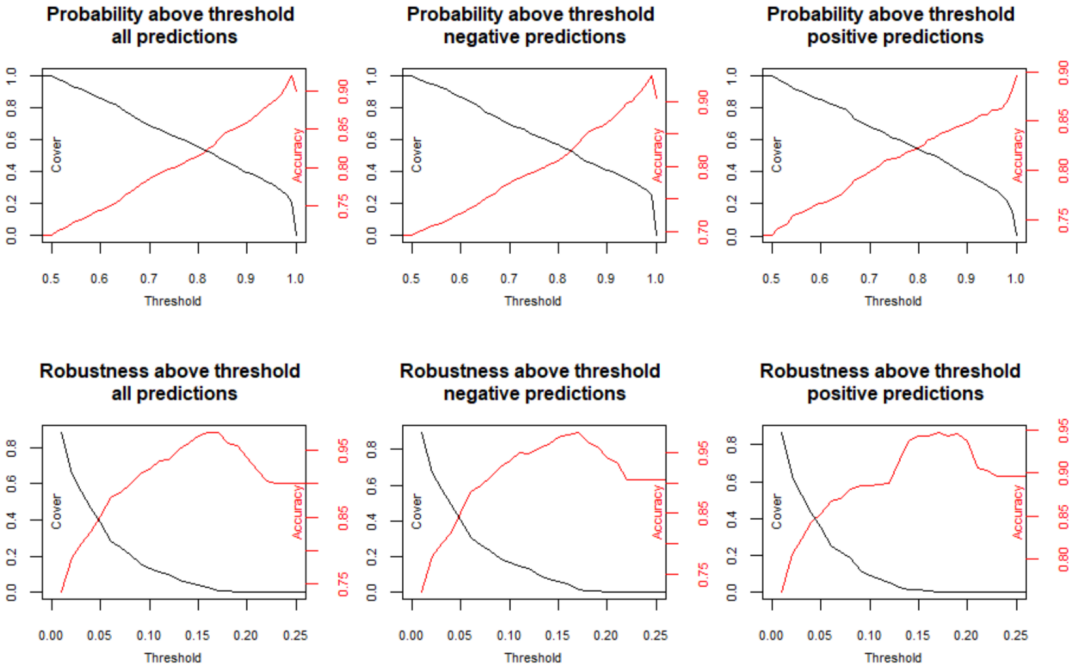


FIGURE 5.1: Cover and accuracy of probability and robustness above threshold in CSPNs: given a threshold for a reliability metric, we can achieve better accuracy and cover of the predictions with robustness than probability.

36,707 orders we described in section 4.1.1, which we used for describing and preparing the data. As described by Mauá et al. (2017), however, CSPNs' benefits are especially pronounced when trained on sparse data. Therefore, the limited size of the dataset used for this analysis exemplifies the applicability of robustness as a reliability metric.

In addition to the number of observations being reduced as compared to the analysis in the three-way comparison, the number of features was also reduced. The 23 features with the highest variance were selected, as well as the continuous `Total_Price` feature.

The SPN was trained using the procedure of Gens and Pedro (2013), with the exception that independence tests were performed using one of Chi-square, Kruskal-Wallis or Kendall (according to the variables involved), clustering was done with the Gower distance (so as to take into account both categorical and continuous variables), and leaves related to continuous variables were forced to be normally distributed.

The robustness value was then calculated through the method described by Mauá et al. (2017). Both robustness and probability of most probable class were used in order to discriminate the quality of the predictions. Figure 5.1 shows the results obtained by issuing a classification only when the associated reliability metric met a certain threshold. Using probability no threshold allowed us to exceed experts' manual performance. Robustness, on the other hand, allowed us to achieve on-par accuracy over 0.15 of all orders.

This analysis showed that robustness was a better indicator of reliability than probability in this context. As we will discuss in the following section, however, it should be noted that there is a difference between experts' overall accuracy and their accuracy over the subset of most robust predictions. We further investigated this nuance in the three-way comparison over the entire dataset.

	Manual	NB	CSPN	XGB
Accuracy	0.93	0.82	0.83	0.88
True negative	0.77	0.73	0.72	0.73
False positive	0.02	0.06	0.06	0.06
False negative	0.05	0.12	0.10	0.05
True positive	0.16	0.11	0.11	0.16

TABLE 5.2: Confusion matrices: how do the various models compare against each other and the experts' review?

	NB	CPSN	XGB
probability	0.60	0.56	0.84
robustness	0.69	0.63	-

TABLE 5.3: Cover for manual accuracy: how many of a classifiers predictions can we accept while achieving the same accuracy that is on par with experts over the entire dataset?

5.2.2 Three-way comparison

Having trained the three different models, we were able to compare their predictive accuracy against one another. Additionally, their accuracy was compared to that of experts' manual review as well. The code used for this comparison can be found in appendix H, with the code for the visualisations being located in appendix I.

None of the models exceeded experts' overall accuracy, as can be seen in table 5.2. Out of the three models the XGB classifier performed best, with 0.88 overall accuracy. Looking at the differences between manual review and the predictions from the XGB classifier, it appears that the latter tends to classify legitimate orders as fraudulent more often. This can in part be explained by the hyper-parameter setting `scale_pos_weight`, where we attribute a higher weight to fraudulent orders. As described in section 2.2 this makes sense business-wise, since false negatives are far more costly than false positives.

The NB and CSPN classifiers, which do not incorporate these different weights, perform somewhat worse. For the NB classifier this is in line with previous research which shows XGB to outperform NB classifiers on a regular basis (Wainer, 2016). The fact that the CSPN classifier does not outperform the XGB classifier can be attributed to the fact that CSPNs' strengths (i.e.: performing well with sparse datasets) are not pronounced in the current context.

Despite achieving lower overall accuracy, the models show promise for creating a better business nevertheless. Accepting each and every prediction the models make at face-value would be too costly. Using their reliability scores to determine which predictions we can trust, however, yields better results. As shown in table 5.3 the models achieve on-par accuracy for part of the dataset. The XGB classifier performs best: if we want on-par overall accuracy, we can achieve a cover of 0.84. For the NB and CSPN classifiers, robustness remains a considerably better estimator for reliability than robustness, just as shown in the previous section.

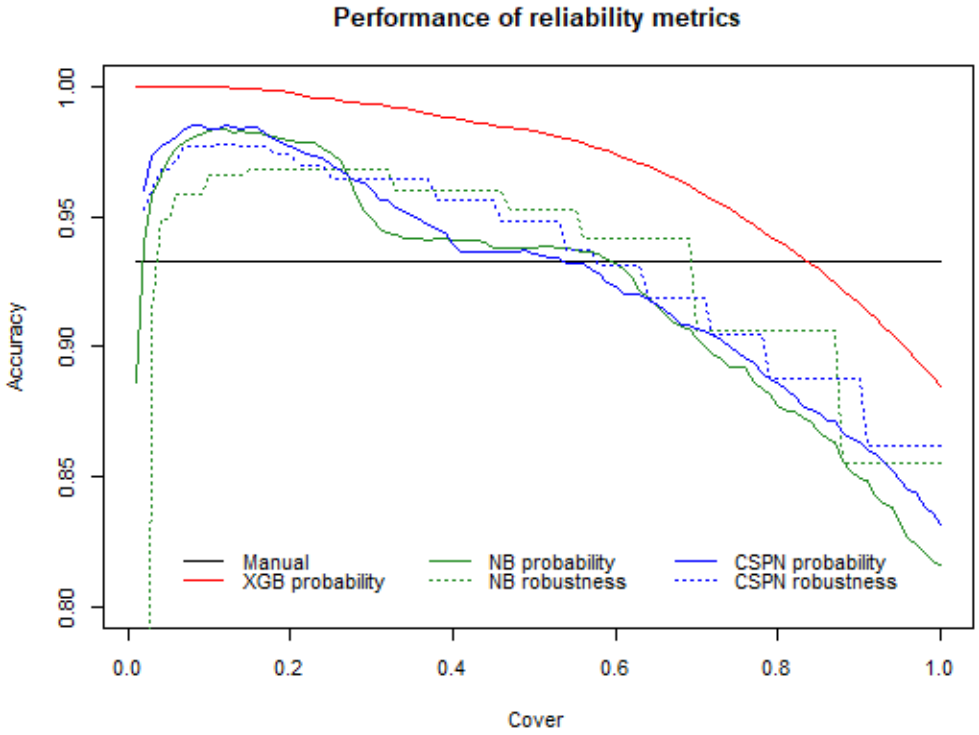


FIGURE 5.2: Performance of reliability metrics: how much cover can be achieved against which accuracy, going by the predictions with the highest reliability scores.

For each of the five model/reliability metric-combinations, we plotted the achieved accuracy against the cover over the dataset. As shown in figure 5.2 all models perform better than experts up to a certain cover. In other words, we could set a threshold for the reliability score and leave instances that do not achieve this threshold up for manual classification. This would allow for both an increase in overall accuracy, as well as a reduced workload.

This approach is slightly naive, however. Or rather, the actual results are somewhat less spectacular than they appear at first. The underlying assumption of this approach is that experts' accuracy is constant over all classifications. While reliability is a latent variable, it plays a part in manual review nonetheless. Simply put: not all orders are equally difficult to classify for experts.

In that case, we should be comparing the classifiers' performance against performance of experts over the exact same subset rather than the entire dataset. Looking at figure 5.3 we can see that the experts' accuracy is indeed not constant over all cases. For the 0.69 instances with the highest reliability score, the XGB classifier does as well as manual review. This is still a promising prospect, but slightly less so than the 0.84 cover we achieved under the previous assumption. The other reliability metrics show a similar pattern: where the model is more reliable, the expert performs better than over the whole dataset. Visualisations of the other classifiers' performance can be found in appendix G.

Knowing this, one could now set a reliability threshold to determine the what portion of the predictions to trust. Effectively, this provides a button by which it is possible to

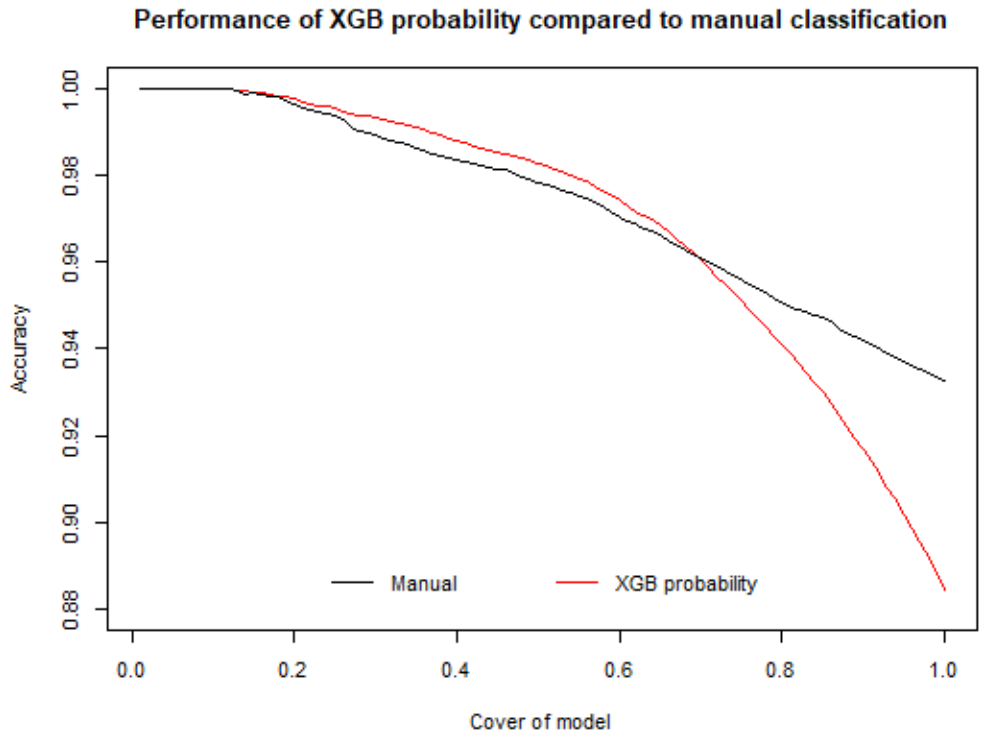


FIGURE 5.3: XGB probability versus manual classification: what is the experts' accuracy over the same observations covered by the model?

determine the trade-off between accuracy and workload. In this context, this might be worth adjusting at times when very many or very few orders are expected by bol.com.

Aside from the outright classification of orders, we can also use the classifiers somewhat differently to improve upon the existing situation. XGB allows for analysis of feature importance, providing insights into which aspects of an order are typical of a fraudulent order. The experts can take these insights into consideration to improve their analytical skills, or use them to improve the business rules that are currently in place. The decision tree of the XGB classifier can also be analysed as a whole, allowing for elicitation of business rules that is independent of domain knowledge.

The same approach can be taken in a slightly less sophisticated manner for each of the reliability metrics. We have determined that observations with a low reliability score are inherently more difficult to classify, even for experts. Further analysis of these difficult cases could allow experts to extract additional features and expand upon the existing business rules.[1]

[1]Because feature importance scores and the decision tree provide a step-by-step guide into how the legitimacy of orders is determined, these insights could put bol.com's business in jeopardy. As such, we cannot publish these concrete findings.

Chapter 6

Discussion

In the following chapter we provide a brief summary of this thesis project, before drawing conclusions and answering our research questions. We then discuss the limitations of this project, and highlight opportunities for future research.

6.1 Summary

Adoption of machine learning models within the industry is lagging. Lack of trust in the predictions generated by models plays a major part in this. Reliable machine learning seeks to mitigate this issue by providing a measure of reliability to predictions.

Throughout this thesis we investigated the applicability of reliable machine learning for high-stakes classifications. Specifically, we conducted a case study into RML for fraud detection at bol.com, the Netherlands' largest online retailer. We analysed orders from between October 2018 and January 2019 in order to to achieve the following research goal:

"This research aims to improve fraud detection by designing a reliable machine learning model that classifies fraudulent orders with acceptable accuracy so that bol.com can increase working efficiency."

While a business rules-based system for identifying fraudulent orders currently exists at bol.com, 0.2%* of those orders are flagged for manual review by an expert. We set out to classify these orders using reliable machine learning models, to see how reliable machine learning models could contribute to existing business processes.

We conducted a data science project following the CRISP-DM method. We collected data from four different sources (order data, fraud function execution data, payment data, and customer support data) and combined these datasets to get one labelled dataset. We divided these datasets into a train- and test-set, and created three different classifiers: a Naive Bayes classifier, a Credal Sum-Product Network, and an XGBoost gradient booster.

The XGB classifier performed best of these three, but none were able to achieve an overall accuracy on-par with manual review. Nevertheless, we were able to subset the predictions based on the models' reliability metrics. Both robustness and probability allowed for this, and we were able to achieve higher prediction than overall manual accuracy over subsets with higher reliability. As such, 0.69 of the XGB classifier's prediction could be relied upon without sacrificing predictive accuracy.

6.2 Conclusions

In this thesis we set out to answer a number of research questions. Having fully conducted this research project, we are now able to answer these questions as follows:

1. **Why are machine learning models not being widely used for high-stakes classifications?** Literature shows that the industry is hesitant to adopt machine learning because of a lack of trust in models' predictions. Reliable machine learning seeks to mitigate this issue by accompanying predictions with a metric that establishes their reliability according to the model.
2. **What metrics can be used to determine the reliability of classifications made by machine learning models?** We can distinguish between model-dependent and -independent reliability metrics. Instances of these metrics are probability and robustness respectively. Both of these metrics are good estimators of reliability, although robustness outperforms probability within the same model.
3. **Which machine learning approaches can be applied for high-stakes classifications?** While we did not conduct an exhaustive review, we found that supervised learning methods that generate a classifier can be used within the domain of fraud detection. Specifically, Naive Bayes classifiers, Credal Sum-Product Networks, and XGBoost Gradient Boosters are applicable to this domain.

Answering these sub-questions contributed to the investigation of our main research question, which we can answer by drawing the following conclusions:

"How can reliable machine learning be applied to improve upon existing business processes for high-stakes classifications?"

RML models can be trained to classify observations of a dataset and provide associated reliability measures such as probability and robustness. In our specific context, we conclude that an RML model can be incorporated in the existing infrastructure of bol.com in order to further their business needs, thus achieving our research goal. Over subsets with a high reliability score a model can achieve on-par or higher accuracy than manual review. This allows for automation of manual labour, scalability benefits, and externalisation of order classification as a service. For bol.com specifically, this would contribute to their strategy of becoming the leading e-commerce platform.

Aside from providing concrete opportunities for bol.com to improve upon their business, this case study also provides general insights into the applicability of RML in high-stakes domains. We both validate and expand upon existing research in a real-world context. The assessment of the models' classifications shows that reliability metrics allow for subsetting of a given model's predictions in such a manner that higher accuracy is achieved over a certain proportion (or *cover*) of the predictions. This illustrates how reliable machine learning models can be implemented to improve upon existing business processes through the following manners:

- Providing accurate predictions over a portion of the observations even when overall predictive accuracy is lower than that of existing approaches.
- Allowing for a flexible accuracy/cover trade-off, where one can accept a decrease in accuracy for an increase in cover or vice-versa.

- Determining which observations are difficult to classify and therewith inferring the latent difficulty variable of those observations.
- Eliciting features and feature combinations that allow for improvement of rules-based systems and experts' analytical skills.

These conclusions apply to the domain as a whole, thus extending the existing scientific knowledge base. We have shown that robustness outperforms probability as a reliability metric for CSPN and NB classifiers in the domain of fraud detection. We furthermore demonstrated the novel capability of CSPNs calculating these measures over datasets with continuous features, which extends their applicability to many new domains.

6.3 Limitations

Even though this project yields tangible contributions to the general knowledge base on RML, it has certain limitations. First of all, the project was conducted in a post-hoc setting, only analysing data for which the eventual outcome was known. Implementation in a live environment —while taken into consideration when selecting data sources— was not studied in and of itself.

Furthermore, while we performed a comparison of three different ML techniques, the comparison was by no means exhaustive. As such, we cannot say with certainty which models and associated reliability metrics perform best under which circumstances. Additionally, little feature selection was performed when training the models. Extensive selection beyond an assessment of variability could improve the overall performance of the models and affect the provided reliability metrics.

Lastly, the classifiers we created were all built upon existing business rules in place at bol.com. While this showcases how RML techniques can be used to used to expand upon existing business processes, it also impacts generalisability beyond the current context. We did not demonstrate how (R)ML models could function as a stand-alone system, and their continued performance in this context necessitates that the various black- and whitelists from which the features are derived are kept up-to-date.

6.4 Future research

The research we conducted leaves a number of questions to be answered, and therewith opportunities for continued research. While the XGB classifier achieved the best performance using the probability metric, the other classifiers showed robustness to be a better indicator of reliability. It would therefore be worth investigating whether incorporating a robustness metric in other models (such as XGB) would be possible and beneficial.

Furthermore, while this research produced a proof of concept, it did not yield a model that was validated in a live environment. Continued research into the deployment of RML models would produce insights into the real-world application of RML models. This would also document the final phase of CRISP-DM, completing the cycle. Studying an RML model in production would also show the interaction effects with the environment in which it is operating.

Lastly, the applicability of this thesis' findings could be studied in another context than within bol.com. This would provide validation to our results and mitigate the generalisability concerns.

Bibliography

[1] Vineeth Balasubramanian, Shen-Shyang Ho, and Vladimir Vovk. *Conformal prediction for reliable machine learning: theory, adaptations and applications*. Newnes, 2014.

[2] Dan Becker. *Learn Machine Learning: what is XGBoost?* 2018. URL: https://www.kaggle.com/dansbecker/xgboost (Retrieved: 23 May 2019).

[3] Zoran Bosnić and Igor Kononenko. "An overview of advances in reliability estimation of individual predictions in machine learning". In: *Intelligent Data Analysis* 13.2 (2009), pp. 385–401.

[4] Meta S. Brown. *CRISP-DM; The dominant process for data mining*. 2015. URL: https://www.youtube.com/watch?v=civLio11SjQ (Retrieved: 22 Nov. 2018).

[5] Meta S. Brown. *What IT Needs To Know About The Data Mining Process*. 2015. URL: https://www.forbes.com/sites/metabrown/2015/07/29/what-it-needs-to-know-about-the-data-mining-process/#2065f3a3515f (Retrieved: 19 Nov. 2018).

[6] Jason Brownlee. *A Gentle Introduction to XGBoost for Applied Machine Learning*. 2016. URL: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/ (Retrieved: 23 May 2019).

[7] Jenna Burrell. "How the machine 'thinks': Understanding opacity in machine learning algorithms". In: *Big Data & Society* 3.1 (2016).

[8] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2015, pp. 1721–1730.

[9] Bojan Cestnik et al. "Estimating probabilities: a crucial task in machine learning." In: *ECAI*. Vol. 90. 1990, pp. 147–149.

[10] Philip K Chan, Wei Fan, Andreas L Prodromidis, and Salvatore J Stolfo. "Distributed data mining in credit card fraud detection". In: *IEEE Intelligent Systems and Their Applications* 14.6 (1999), pp. 67–74.

[11] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rudiger Wirth. *CRISP-DM 1.0 Step-by-step data mining guide*. 2000.

[12] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM. 2016, pp. 785–794.

[13] Diarmaid Conaty, Jesús Martínez Del Rincon, and Cassio P De Campos. "Cascading Sum-Product Networks using Robustness". In: *International Conference on Probabilistic Graphical Models*. 2018, pp. 73–84.

[14] Council of the European Union. *General Data Protection Regulation article 22: Automated individual decision-making, including profiling*. 2016. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1528874672298&uri=CELEX%5C%3A32016R0679 (Retrieved: 21 Nov. 2018).

[15] Rob C. De Wit, Cassio P. De Campos, Diarmaid Conaty, and Jésus Marínez Del Rincon. "Robustness in Sum-Product Networks with Continuous and Categorical Data". In: *Proceedings of the 2019 International Symposium on Imprecise Probabilities*. 2019.

[16] Deloitte. *Machine learning: things are getting intense*. 2017. URL: https://www2.deloitte.com/content/dam/Deloitte/global/Images/infographics/technologymediatelecommunications/gx-deloitte-tmt-2018-intense-machine-learning-report.pdf (Retrieved: 21 Nov. 2018).

[17] Tom Fawcett and Foster J Provost. "Combining Data Mining and Machine Learning for Effective User Profiling." In: *KDD*. 1996, pp. 8–13.

[18] Yoav Freund, Robert Schapire, and Naoki Abe. "A short introduction to boosting". In: *Journal-Japanese Society For Artificial Intelligence* 14.771-780 (1999), p. 1612.

[19] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[20] Alexander Gammerman and Vladimir Vovk. "Hedging predictions in machine learning". In: *The Computer Journal* 50.2 (2007), pp. 151–163.

[21] Robert Gens and Domingos Pedro. "Learning the structure of sum-product networks". In: *International conference on machine learning*. 2013, pp. 873–880.

[22] Diana Gordon and Donald Perlis. "Explicitly biased generalization". In: *Computational Intelligence* 5.2 (1989), pp. 67–81.

[23] Arthur D. Hall. *A methodology for systems engineering*. van Nostrand, 1962.

[24] Howard J Hamilton, Ning Shan, and Wojciech Ziarko. "Machine learning of credible classifications". In: *Australian Joint Conference on Artificial Intelligence*. Springer. 1997, pp. 330–339.

[25] J Christopher Jones. *Design Methods: Seeds of human futures*. 1980.

[26] Jeamin Koo, Jungsuk Kwac, Wendy Ju, Martin Steinert, Larry Leifer, and Clifford Nass. "Why did my car just do that? Explaining semi-autonomous driving actions to improve driver understanding, trust, and performance". In: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 9.4 (2015), pp. 269–275.

[27] Miroslav Kubat, Stan Matwin, et al. "Addressing the curse of imbalanced training sets: one-sided selection". In: *Icml*. Vol. 97. Nashville, USA. 1997, pp. 179–186.

[28] Matjaž Kukar and Igor Kononenko. "Reliable classifications with machine learning". In: *European Conference on Machine Learning*. Springer. 2002, pp. 219–231.

[29] Pat Langley. "The changing science of machine learning". In: *Machine Learning* 82.3 (2011), pp. 275–279.

[30] Monkol Lek, Benjamin Anadarajah, Narciso Cerpa, and Rodger Jamieson. "Data mining prototype for detecting ecommerce fraud". In: *ECIS 2001 Proceedings* (2001), p. 60.

[31] Ben Lorica and Paco Nathan. *The State of Machine Learning Adoption in the Enterprise*. O'Reilly, 2018. URL: https://www.oreilly.com/data/free/state-of-machine-learning-adoption-in-the-enterprise.csp (Retrieved: 16 Nov. 2018).

[32] V. Maini and S. Sabri. *Machine Learning for Humans*. 2017. URL: https://medium.com/machine-learning-for-humans/ (Retrieved: 16 Jan. 2019).

[33] Melvin Earl Maron. "Automatic indexing: an experimental inquiry". In: *Journal of the ACM (JACM)* 8.3 (1961), pp. 404–417.

[34] Denis D. Mauá, Fabio G. Cozman, Diarmaid Conaty, and Cassio P. Campos. "Credal sum-product networks". In: *Proceedings of the Tenth International Symposium on Imprecise Probability: Theories and Applications*. 2017, pp. 205–216.

[35] Mehryar Mohri, Ameet Talwalkar, and Afshin Rostamizadeh. *Foundations of machine learning (adaptive computation and machine learning series)*. Mit Press Cambridge, MA, 2012.

[36] Frank Pasquale. *The black box society: The secret algorithms that control money and information*. Harvard University Press, 2015.

[37] Manish Pathak. *Using XGBoost in Python*. 2018. URL: `https://www.datacamp.com/community/tutorials/xgboost-in-python` (Retrieved: 23 May 2019).

[38] Michael Pazzani, Christopher Merz, Patrick Murphy, Kamal Ali, Timothy Hume, and Clifford Brunk. "Reducing misclassification costs". In: *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 217–225.

[39] Hoifung Poon and Pedro Domingos. "Sum-product networks: A new deep architecture". In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE. 2011, pp. 689–690.

[40] Félix Revert. *Fine-tuning XGBoost in Python like a boss*. 2018. URL: `https://towardsdatascience.com/fine-tuning-xgboost-in-python-like-a-boss-b4543ed8b1e` (Retrieved: 18 Apr. 2019).

[41] Stefan Rüping et al. "Learning interpretable models". PhD dissertation. Dortmund University, 2006.

[42] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[43] Arthur L Samuel. "Some studies in machine learning using the game of checkers". In: *IBM Journal of research and development* 3.3 (1959), pp. 210–229.

[44] Craig Saunders, Alexander Gammerman, and Volodya Vovk. "Transduction with confidence and credibility". In: *Sixteenth International Joint Conference on Artificial Intelligence*. 1999, pp. 722–726.

[45] Signifyd and PYMNTS. *Global Fraud Index Q2 2017*. 2017. URL: `https://www.pymnts.com/global-fraud-index/` (Retrieved: 16 Nov. 2018).

[46] Bruno Stecanella. *A practical explanation of a Naive Bayes classifier*. 2017. URL: `https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/` (Retrieved: 21 June 2019).

[47] Salvatore Stolfo, David W Fan, Wenke Lee, Andreas Prodromidis, and P Chan. "Credit card fraud detection using meta-learning: Issues and initial results". In: *AAAI-97 Workshop on Fraud Detection and Risk Management*. 1997.

[48] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.

[49] Jacques Wainer. "Comparison of 14 different families of classification algorithms on 115 binary datasets". In: *arXiv preprint arXiv:1606.00930* (2016).

[50] Sida Wang and Christopher D Manning. "Baselines and bigrams: Simple, good sentiment and topic classification". In: *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*. Association for Computational Linguistics. 2012, pp. 90–94.

[51] Andreas S Weigend and David A Nix. "Predictions with confidence intervals (local error bars)". In: *Proceedings of the international conference on neural information processing*. 1994, pp. 847–852.

[52] Roel J Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.

[53] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

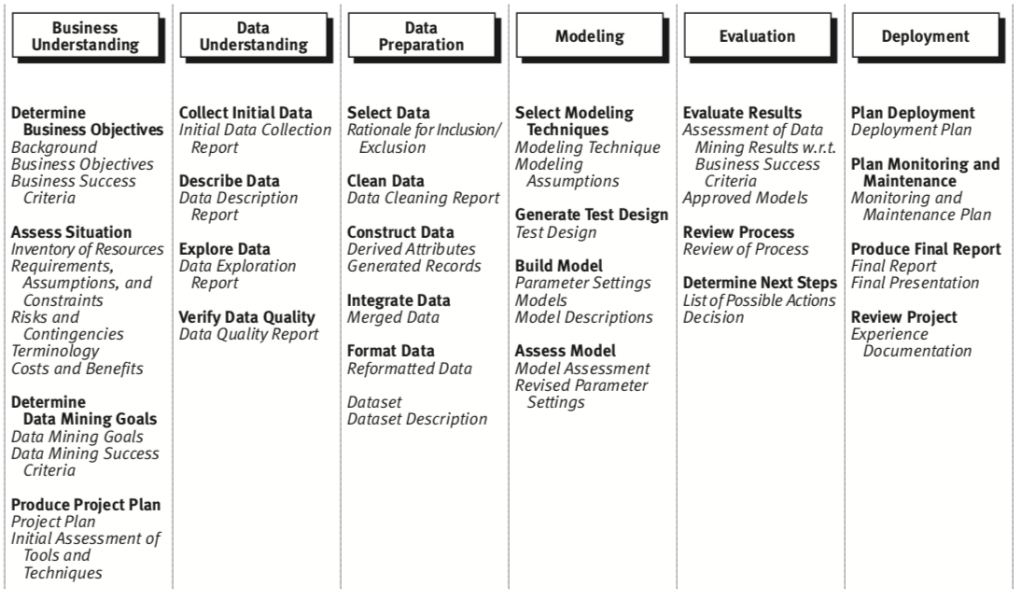
[54] Keith Wong, Brenda Ng, Narciso Cerpa, and Rodger Jamieson. "An Online Audit Review System for Electronic Commerce". In: *Proceedings of the 13th Bled Electronic Commerce Conference*. 2000, pp. 20–23.

[55] Masoumeh Zareapoor and Pourya Shamsolmoali. "Application of credit card fraud detection: Based on bagging ensemble classifier". In: *Procedia Computer Science* 48.2015 (2015), pp. 679–685.

Appendix A

Generic tasks CRISP-DM

A schematic overview of the generic tasks (bold) and their outputs (italic) by phase in CRISP-DM (Chapman et al., 2000):



Appendix B

R code data pre-processing

```
library(tidyr)
library(stringr)
library(splitstackshape)
library(onehot)
library(mltools)
library(data.table)

# Import the datasets
# Separate months have separate CSVs, hence the binding
raw_functions <- read.csv2(file.choose(), sep = ",")
raw_functions <- rbind(raw_functions, read.csv2(file.choose()
    ↪ , sep = ","))

raw_payment <- read.csv2(file.choose())
raw_payment <- rbind(raw_payment, read.csv2(file.choose()))

raw_csupport <- read.csv2(file.choose(), sep = ",")
raw_csupport <- rbind(raw_csupport, read.csv2(file.choose(),
    ↪ sep = ","))

raw_orderdata <- read.csv2(file.choose(), sep = ";")
raw_orderdata <- rbind(raw_orderdata, read.csv2(file.choose()
    ↪ , sep = ";"))

# Get a list of all functions and their output types
functions <-unique(raw_functions[, c("Name", "functionType")
    ↪ ])

# Get a list of unique orders and their associated function
    ↪ outputs
# Make a copy of the raw data
functionExecutions <- raw_functions
functionExecutions$Name <- paste(functionExecutions$Name,
    ↪ functionExecutions$functionType, sep = "-")

# Delete function executions with differing results
# Some functions return a different value depending on the
    ↪ parameter, but we don't have the parameters. Therefore
    ↪ these functions don't provide any info to us.
```

```
functions_double_outputs <- functionExecutions[, c("
    ↪ orderReference", "Name", "Result")]
functions_double_outputs <- unique(functions_double_outputs)
functions_double_outputs <- functions_double_outputs[
    ↪ duplicated(functions_double_outputs[, 1:2]) |
    ↪ duplicated(functions_double_outputs[, 1:2], fromLast =
    ↪ TRUE), ]
functions_double_outputs <- unique(functions_double_outputs[,
    ↪  "Name"])

# Remove these uncertain results from our dataset
functionExecutions <- functionExecutions[!functionExecutions$
    ↪ Name %in% functions_double_outputs, ]
functionExecutions[functionExecutions == "true"] <- TRUE
functionExecutions[functionExecutions == "false"] <- FALSE

# Combine rows to have Function Outputs as columns with Order
    ↪ _ID as a key
functionExecutions <- functionExecutions[,c("orderReference",
    ↪  "Name", "Result")]
functionExecutions <- unique(functionExecutions)
functionExecutions <- reshape(functionExecutions, idvar = "
    ↪ orderReference", timevar = "Name", direction = "wide")
colnames(functionExecutions)[1] <- "ORDER_ID"

# Data types
functionExecutions[,2:ncol(functionExecutions)] <- sapply(
    ↪ functionExecutions[,2:ncol(functionExecutions)], as.
    ↪ character)
functionExecutions[,2:ncol(functionExecutions)] <- sapply(
    ↪ functionExecutions[,2:ncol(functionExecutions)], as.
    ↪ logical)

# Add the orderdata, change their names for ease of use.
orders <- raw_orderdata
orders[, c("Ind.fraud", "Orders..Dist....")] <- NULL
colnames(orders)[2] <- "ORDER_ID"
colnames(orders)[3] <- "CITY_SHIPMENT"
colnames(orders)[4] <- "ACCOUNT_TYPE"
colnames(orders)[5] <- "PICKUP_LOCATION"
colnames(orders)[6] <- "CANCELLATION_REASON"
colnames(orders)[7] <- "GPC"
colnames(orders)[8] <- "TOTAL_PRICE"
colnames(orders)[9] <- "DELIVERY_METHOD"
colnames(orders)[10] <- "CUSTOMER_ID"
colnames(orders)[11] <- "DATE_FIRST_ORDER"
colnames(orders)[12] <- "GENDER"
colnames(orders)[13] <- "DATE_OF_BIRTH"
colnames(orders)[14] <- "PAYMENT_METHOD"
colnames(orders)[15] <- "SELLER_ID"
colnames(orders)[16] <- "SELLER_TYPE"
```

```
# Convert to proper datatypes and clean up some data
orders$DATE_OF_BIRTH <- as.Date(orders$DATE_OF_BIRTH, "%d-%m
    ↪ -%Y")
orders$DATE_FIRST_ORDER <- as.Date(orders$DATE_FIRST_ORDER, "
    ↪ %d-%m-%Y")

# Deleted accounts ("forgotten")
# Data from these accounts was scrambled. Change these values
    ↪  to NA to reflect this.

for(i in 1:nrow(orders)) {
  gender <- orders[i, "GENDER"]
  dfo <- orders[i, "DATE_FIRST_ORDER"]

  if(dfo == "9999-09-09" | is.na(dfo)){orders[i, "DATE_FIRST_
      ↪ ORDER"] <- NA}
  if((gender != "M" & gender != "F") | is.na(gender)){
    orders[i, "GENDER"] <- NA

    #Deal with removed accounts (scrambled personal data)
    if(orders[i, "DATE_OF_BIRTH"] == "1900-01-01" | is.na(
        ↪ orders[i, "DATE_OF_BIRTH"])){
      orders[i, "DATE_OF_BIRTH"] <- NA
      orders[i, "CITY_SHIPMENT"] <- NA
    }
  }

  if(orders[i, "SELLER_ID"] == -1 | is.na(orders[i, "SELLER_
      ↪ ID"])){
    orders[i, "SELLER_ID"] <- NA
  }

  if(orders[i, "PICKUP_LOCATION"] == "NONE" | is.na(orders[i,
      ↪  "PICKUP_LOCATION"])){
    orders[i, "PICKUP_LOCATION"] <- NA
  }
}
orders$GENDER <- droplevels(orders$GENDER)
orders$CITY_SHIPMENT <- droplevels(orders$CITY_SHIPMENT)
rm(gender, dfo, i)


# And now we merge the function results with the payment data
payment <- raw_payment

colnames(payment)[1] <- "ORDER_ID"
colnames(payment)[2] <- "OPEN"
colnames(payment)[3] <- "WO_FRAUD"
colnames(payment)[4] <- "TOTAL_PRICE"
colnames(payment)[5] <- "WO_CHARGEBACK"
```

```
colnames(payment)[6] <- "WO_PHISHING"
colnames(payment)[7] <- "REMINDER_LEVEL"

payment[, c(1:7)] <- sapply(payment[, c(1:7)], as.character)
payment[, c(1:7)] <- sapply(payment[, c(1:7)], as.numeric)

# We don't really care why a write-off was made, so we
    ↪ combine these into a  single feature. Loss is WO + not
    ↪ paid.
payment$WO_TOTAL <- payment$WO_FRAUD + payment$WO_CHARGEBACK
    ↪ + payment$WO_PHISHING
payment$TOTAL_LOSS <- payment$WO_TOTAL + payment$OPEN

# Get a list of ORDER_IDs and whether they have been paid for
    ↪ .
paymentUnpaid <- payment[, c("ORDER_ID", "TOTAL_PRICE", "
    ↪ TOTAL_LOSS", "WO_PHISHING")]
paymentUnpaid$UNPAID <- paymentUnpaid$TOTAL_LOSS > 0
paymentUnpaid$ACCOUNT_TAKEOVER <- paymentUnpaid$WO_PHISHING >
    ↪  0
paymentUnpaid <- paymentUnpaid[, c(1, 5, 6)]

# Match order data with payment data
orders <- merge(orders, paymentUnpaid, by = "ORDER_ID", all =
    ↪  TRUE)

# Combine CSupport data with rest
# We ditch all information except for "has this customer
    ↪ contacted CS?"
# Reasons are not important.

csupport <- raw_csupport[, 1:2]
colnames(csupport)[1] <- "ORDER_ID"
colnames(csupport)[2] <- "CONTACTED_CS"
csupport[,2] <- TRUE
orders$CONTACTED_CS <- orders$ORDER_ID %in% csupport$ORDER_ID

# Include verdict of fraud: did the analyst label it as such?
orders$VERDICT_ANALYST <- orders$CANCELLATION_REASON %in% c("
    ↪ FRAUD_BE", "FRAUD_CS")

# Drop rows where ORDER_ID is NA (some silly cleaning)
orders <- orders[-which(is.na(orders[, "ORDER_ID"])),]
    ↪                              # exclude rows which are empty
orders <- orders[-which(is.na(orders[, "ACCOUNT_TYPE"])),]
    ↪                          # exclude rows without order data

# Combine with fraud function execution data
orders <- merge(orders, functionExecutions, by = "ORDER_ID")

# Analyse where we expect accounts to have been taken over
```

```
# This affects how we perceive a CS case: takeover customers
    ↪ tend to contact Bol.com even if their order was
    ↪ rightfully cancelled.
orders$ACCOUNT_TAKEOVER <- orders$ACCOUNT_TAKEOVER | (orders$
    ↪ `Result.OrderApplicantEmailDiffersPreviousOrder-
    ↪ blacklist` & orders$`Result.
    ↪ OrderIpDiffersFromKnownOrders-blacklist`)

# Labeling: Determine whether an order was actually
    ↪ fraudulent

labelFraud <- function(row){
  # If the analyst says it's fraudulent...
  if(row$VERDICT_ANALYST){
    # Return false if CS was contacted, unless it's related
        ↪ to account takeover
    if((is.na(row$ACCOUNT_TAKEOVER) | row$ACCOUNT_TAKEOVER ==
        ↪  FALSE)){
      return(!row$CONTACTED_CS)
    } else {
      return(TRUE)
    }
    # If the analyst approved the order, return TRUE if it
        ↪ wasn't paid for
  } else if(is.na(row$UNPAID) | !row$UNPAID){
    return(FALSE)
  } else if(row$UNPAID){
    return(TRUE)
  }

  return(NA)
}

# Loop the function over all orders. 0 points for style.
for(i in 1:length(orders[, 1])){
  orders[i, "IS_FRAUD"] <- labelFraud(orders[i, ])
}
rm(i)

# Some last formatting
orders[which(orders$DELIVERY_METHOD == ""), "DELIVERY_METHOD"
    ↪ ] <- NA
orders$CITY_SHIPMENT <- toupper(orders$CITY_SHIPMENT)
orders$CITY_SHIPMENT <- gsub("[^A-Z]", "", orders$CITY_
    ↪ SHIPMENT)
orders[which(orders$CITY_SHIPMENT == ""), "CITY_SHIPMENT"] <-
    ↪  NA

orders <- droplevels(orders)
```

```
# Select the data which can be used for training and testing
    ↪ of the model.
output <- orders[which(orders$CANCELLATION_REASON != "FRAUD_
    ↪ BE"), c(2:5, 7:16, 21:76)]

# We now have all features that could be used for training
    ↪ and testing.
# There are, however, a few features left we don't want to
    ↪ use (or aren't available in the cloud as of now)
# So we remove these features. As it turns out, they don't
    ↪ have a huge impact on accuracy anyway.
output[, c("Yrmonth", "CITY_SHIPMENT", "PICKUP_LOCATION", "
    ↪ GPC", "CUSTOMER_ID", "SELLER_ID", "GENDER", "DATE_FIRST
    ↪ _ORDER", "DATE_OF_BIRTH")] <- NULL
output$ACCOUNT_TYPE <- output$ACCOUNT_TYPE == "B"

# Remove features with little variation and convert to 1s and
    ↪  0s.
output[, nearZeroVar(output, freqCut = 100/0)] <- NULL
#output <- as.data.frame(one_hot(as.data.table(output), cols
    ↪ = c(3:39), naCols = TRUE, dropCols = TRUE))
output <- as.data.frame(predict(onehot(output,
    ↪ stringsAsFactors = TRUE, max_levels = 30, addNA = TRUE)
    ↪ , output))

output$VERDICT_ANALYST <- orders[which(orders$CANCELLATION_
    ↪ REASON != "FRAUD_BE"), "VERDICT_ANALYST"]

# Create stratified train and test sets
temp <- stratified(output, "IS_FRAUD", .8, bothSets = TRUE)
train_data <- as.data.frame(temp[[1]])
test_data <- as.data.frame(temp[[2]])

resMAN <- test_data[, c("IS_FRAUD", "VERDICT_ANALYST")]
train_data$VERDICT_ANALYST <- NULL
test_data$VERDICT_ANALYST <- NULL

write.csv2(resMAN, file = "resultsManual-noBE.csv")
write.csv2(train_data, file = "trainset-noBE.csv")
write.csv2(test_data, file = "testset-noBE.csv")
```

Appendix C

Feature distributions of dataset

feature	FALSE	TRUE	NA	feature	FALSE	TRUE	NA
f1	192703	7236	0	**f22**	38211	73929	87799
f2.1	199086	853	0	**f23**	119585	7032	73322
f2.2	184718	15221	0	**f24**	23204	957	175778
f2.3	17979	181960	0	**f25**	179795	7643	12501
f2.4	197412	2527	0	**f26**	199861	78	0
f2.5	166226	33713	0	**f27**	58561	440	140938
f2.6	189176	10763	0	**f28**	11181	1609	187149
f2.7	190754	9185	0	**f29**	186524	7939	5476
f2.8	199812	127	0	**f30**	13725	1943	184271
f2.9	199877	62	0	**f31**	199732	207	0
f2.10	56379	143560	0	**f32**	23873	14732	161334
f2.11	199937	2	0	**f33**	28770	28857	142312
f3.1	177031	22908	0	**f34**	158934	14638	26367
f3.2	70706	129233	0	**f35**	21618	38604	139717
f3.3	197328	2611	0	**f36**	170570	2604	26765
f3.4	154752	45187	0	**f37**	90180	8567	101192
f4	121402	71298	7239	**f38**	3503	1854	194582
f5	190832	9107	0	**f39**	19126	9035	171778
f6	192417	32	7490	**f40**	1815	811	197313
f7	185313	2013	12613	**f41**	7088	162	192689
f8	183971	391	15577	**f42**	27164	2578	170197
f9	96451	10946	92542	**f43**	1512	1865	196562
f10	199353	20	566	**f44**	166030	16121	17788
f11	192315	164	7460	**f45**	199883	56	0
f12	199936	3	0	**f46**	172121	27809	9
f13	56231	9036	134672	**f47**	199725	214	0
f14	34037	1	165901	**f48**	75710	96155	28074
f15	139803	60034	102	**f49**	566	193077	6296
f16	197139	2457	343	**f50**	3437	110323	86179
f17	189685	2764	7490	**f51**	199877	62	0
f18	23988	14620	161331	**f52**	6244	174248	19447
f19	199898	41	0	**f53**	22889	1241	175809
f20	164627	8160	27152	**f54**	64105	18009	117825
f21	199244	566	129				

Appendix D

R code training XGB

```
library(dplyr)
library(xgboost)
library(caret)
library(splitstackshape)

dtrain <- xgb.DMatrix(data = as.matrix(train_data[, -ncol(
    ↪ train_data)]), label= train_data[, ncol(train_data)])
dtest <- xgb.DMatrix(data = as.matrix(test_data[, -ncol(test_
    ↪ data)]), label= test_data[, ncol(test_data)])
test_labels <- test_data[, ncol(test_data)]

# Train model
model <- xgboost(data = dtrain,
                 objective = "binary:logistic",
                 nround = 1000,
                 eta = 0.01,
                 max.depth = 20,
                 min_child_weight = 10,
                 early_stopping_rounds = 100,
                 scale_pos_weight = sum(train_labels == FALSE
                     ↪ )/sum(train_labels == TRUE),
                 #alpha = 3,
                 lambda = 2,
                 #gamma = 5,
                 verbose = 1,
                 missing = -2147483648,
                 #num_parallel_tree = 10,
                 #colsample_bytree = 0.8,
                 print_every_n = 50
)

# Generate predictions for our held-out testing data
pred <- predict(model, dtest)


# Get & print the classification error
accuracy_model <- 1 - mean(as.numeric(pred > .5) != test_
    ↪ labels)
print(accuracy_model)
```

```
write.csv2(pred, file = "resultsXGBoost-noBE.csv")



# Everything below here is just to visualise performance:

# Plot a matrix showing threshold and accuracy for every
    ↪ threshold
# E.g.: if the model classifies everything where p < .25 | p
    ↪ > .75, its cover is already .85 with an accuracy of .97

leaveOutMiddle <- function(threshold){
  pred_certain <- vector(mode="numeric", length=0)
  labels_certain <- vector(mode="numeric", length=0)

  for(n in 1:length(pred)){
    val <- pred[n]
    if(val < threshold | val > (1-threshold)){
      pred_certain <- c(pred_certain, val)
      labels_certain <- c(labels_certain, test_labels[n])
    }
  }
  accuracy_model_certain <- 1 - mean(as.numeric(pred_certain
     ↪ > .5) != labels_certain)
  work_done_by_model <- (length(pred_certain) / length(pred))

  return(c(threshold, work_done_by_model, accuracy_model_
     ↪ certain))
}

# level_of_detail: 100 for 1% steps, 1000 for .1% steps
level_of_detail = 100
certainty_by_threshold <- matrix(data = 0,nrow = level_of_
    ↪ detail/2, ncol = 3)
colnames(certainty_by_threshold) <- c("OuterBounds", "
    ↪ WorkloadHandledByModel", "PredictionAccuracy")

for(n in 1:nrow(certainty_by_threshold)){
  res <- leaveOutMiddle(n/(level_of_detail))
  certainty_by_threshold[n,] <- res
}

# Plot the outcome
par(mar=c(5,4,4,5) + 0.1)
plot(certainty_by_threshold[,1], certainty_by_threshold[,2],
    ↪ type="l", xlab="Outer threshold", ylab = "Work handled
    ↪ by model")
par(new = T)
```

```
plot(certainty_by_threshold[,1], certainty_by_threshold[,3],
    ↪ type="l", xlab=NA, ylab = NA, axes = FALSE, col = 'red'
    ↪ )
axis(side = 4, col.axis = 'red', col.ticks = 'red')
mtext(side = 4, 'Accuracy of predictions', line = 3, col = '
    ↪ red')
abline(h = 0.94, col = 'blue', lty=2)

rm(level_of_detail, res, n)


# Plot feature importance
importance_matrix <- xgb.importance(colnames(train_data),
    ↪ model = model)
xgb.plot.importance(importance_matrix, rel_to_first = TRUE,
    ↪ xlab = "Relative feature importance XGB classifier",
    ↪ top_n = 10, cex = .7)

xgb.plot.tree(model = model)
```

Appendix E

R code training NB and SPN

```
require(igraph)
require(cluster)
require(ROCR)
require(caret)
require(onehot)
require(doParallel)
require(dplyr)

source('spn.value.r')
source('spn.value.int.r')
source('spn.evidence.r')
source('spn.evidence.int.r')
source('spn.learn.r')
source('spn.print.r')
source('spn.test.r')
source('spn.utils.r')

Train <- read.csv2(file.choose(), sep = ";")
Test <- read.csv2(file.choose(), sep = ";")
Train$X <- NULL
Test$X <- NULL

All <- rbind(Train, Test)

All$TOTAL_PRICE <- as.numeric(as.character(All$TOTAL_PRICE))
All$TOTAL_PRICE <-(All$TOTAL_PRICE-min(All$TOTAL_PRICE))/(max
    ↪ (All$TOTAL_PRICE-min(All$TOTAL_PRICE)))

All[All == TRUE] <- 1
All[All == FALSE] <- 0

for(i in 1:ncol(All)){
  All[which(All[, i] == -2147483648), i] <- median(All[which(
      ↪ All[, i] > -1), i])
}

All <- All%>%select(-ACCOUNT_TYPE,ACCOUNT_TYPE)
All <- All%>%select(-IS_FRAUD, IS_FRAUD)
```

```
All[, 2:ncol(All)] <- All[, 2:ncol(All)] + 1

All[, nearZeroVar(All, freqCut = 99/1)] <- NULL
All <- as.matrix(All)

#outputSPN <- spn.cv(Train, nfolds=2, classcol=ncol(Train),
    ↪ verb = TRUE, height = 2)
#cat('pr   ', mean(res[,3]),      mean(res[,1]==res[,2]),
    ↪ mean(res[res[,1]==res[,2],3]),      mean(res[res[,1]!=res
    ↪ [,2],3]), sum(res[,1]!=res[,2]), sum(res[,1]==res[,2]),
    ↪   '\n')
#cat('rob  ', mean(res[,4]),      mean(res[,1]==res[,2]),
    ↪ mean(res[res[,1]==res[,2],4]),      mean(res[res[,1]!=res
    ↪ [,2],4]), '\n')

# Train
summ <- spn.learncats(All, classcol=ncol(All))
spn3 <- spn.learn(summ$data[-(nrow(Train)+1:nrow(All)),], ncat
    ↪ =summ$ncat, maxv=summ$maxv, minv=summ$minv, verb=TRUE,
    ↪ classcol=ncol(All), thr = 0.01)
spnNB3 <- spn.learn(summ$data[-(nrow(Train)+1:nrow(All)),],
    ↪ ncat=summ$ncat, maxv=summ$maxv, minv=summ$minv, verb=TRUE,
    ↪ classcol=ncol(All), thr=0.01, height=2)

# Test
resNB3 <- as.data.frame(spn.predict(spnNB3, All
    ↪ [159952:199939,,drop=FALSE], classcol = ncol(All), run.
    ↪ rob=TRUE, ncores=4,tfname="tfname"))
resSPN3 <- as.data.frame(spn.predict(spn3, All
    ↪ [159952:199939,,drop=FALSE], classcol = ncol(All), run.
    ↪ rob=TRUE, ncores=4,tfname="tfname"))
```

Appendix F

ISIPTA 2019 paper submission

The following short paper was published as Rob C. De Wit et al. "Robustness in Sum-Product Networks with Continuous and Categorical Data". In: *Proceedings of the 2019 International Symposium on Imprecise Probabilities*. 2019.

ISIPTA 2019

Robustness in Sum-Product Networks with Continuous and Categorical Data

Rob C. de Wit R.C.DEWIT@UU.NL
Cassio P. de Campos C.DECAMPOS@UU.NL
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Diarmaid Conaty DCONATY01@QUB.AC.UK
Jesús Martínez del Rincon J.MARTINEZ-DEL-RINCON@QUB.AC.UK
Centre for Data Science and Scalable Computing, Queen's University Belfast, U.K.

Abstract

Sum-product networks are a popular family of probabilistic graphical models for which marginal inference can be performed in polynomial time. After learning sum-product networks from scarce data, small variations of parameters could lead to different conclusions. We adapt the robustness measure created for categorical credal sum-product networks to domains with both continuous and categorical variables. We apply this approach to a real-world dataset of online purchases where the goal is to identify fraudulent cases. We empirically show that such credal models can better discriminate between easy and hard instances than simply using the probability of the most probable class.

Keywords: Robustness, Sum-Product Networks

1. Introduction

Sum-Product Networks (SPNs) are a class of probabilistic graphical models that allow for the explicit representation of context-specific independence [9] while retaining efficient marginal inference [7, 10]. An SPN encodes an arithmetic circuit [3]: internal nodes perform (weighted) sums and multiplications, while leaves represent variable assignments (or marginal distributions of continuous variables). SPNs can be seen as a class of mixture of univariate distributions with tractable inference [4, 8, 11].

SPNs learned from data may generalise poorly and produce unreliable and overconfident conclusions. When variables are categorical, *Credal Sum-Product Networks* (CSPNs), a class of imprecise probability models, can be used to perform a (computationally efficient) robustness analysis of SPNs for classification [1, 2, 5, 6]. However, often real-world data comes with both discrete and continuous variables, which can be used to infer an SPN. We extend CSPNs towards domains with continuous variables. A CSPN is an SPN where the weights associated with sum nodes (i.e., the numerical parameters of the model) are allowed to vary inside a closed and convex set. Continuous variables are represented in leaf nodes and are assumed to be normally distributed. An experimental analysis is conducted using data from a major online retailer, where the goal is to discriminate between fraudulent and legitimate orders. This is a multi-million market and frauds can be very costly.

2. Continuous and Categorical CSPNs

The evaluation of an SPN (i.e., the computation of its value) for a given configuration of variables can be performed by a bottom-up message propagation scheme whereby each node sends to its parent its value. Leaf nodes send a density value (continuous variables) or the result of the indicator function (categorical variables). The whole procedure takes linear time and space. Conditional probabilities for categorical variables can be obtained in linear time by evaluating the network for each value of the query variable and the given evidence (then normalising the result). For CSPNs, more intricate algorithms have been devised to compute the expectation of any function over a single categorical variable. They can be promptly adapted to handle continuous variables, since the propagation of density values is similar to the propagation of probability values. In particular, SPNs (and their inferences) do not need to be normalised, so one needs simply to take the continuous leaf nodes and compute their density values, and then to "send" these values to their parents in the SPN. This is the only required adaptation, while the procedure in the internal nodes remains the same as for categorical CSPNs and the algorithms for credal classification work just as designed before in the literature [5]. In fact, this result can be proven by realising that observed continuous variables act similarly to an observed binary categorical variable, and so we obtain the following theorem.

Theorem 1 *Computing lower conditional expectations of a function over a single categorical variable in* CSPN*s with both categorical and continuous variables takes at most polynomial time when each internal node has at most one parent.*

Because of that, credal classification (i.e., computing the set of non-dominated classes) can be done in polynomial time too. On par with previous work [5], we use CSPNs as means to define the robustness level of an issued

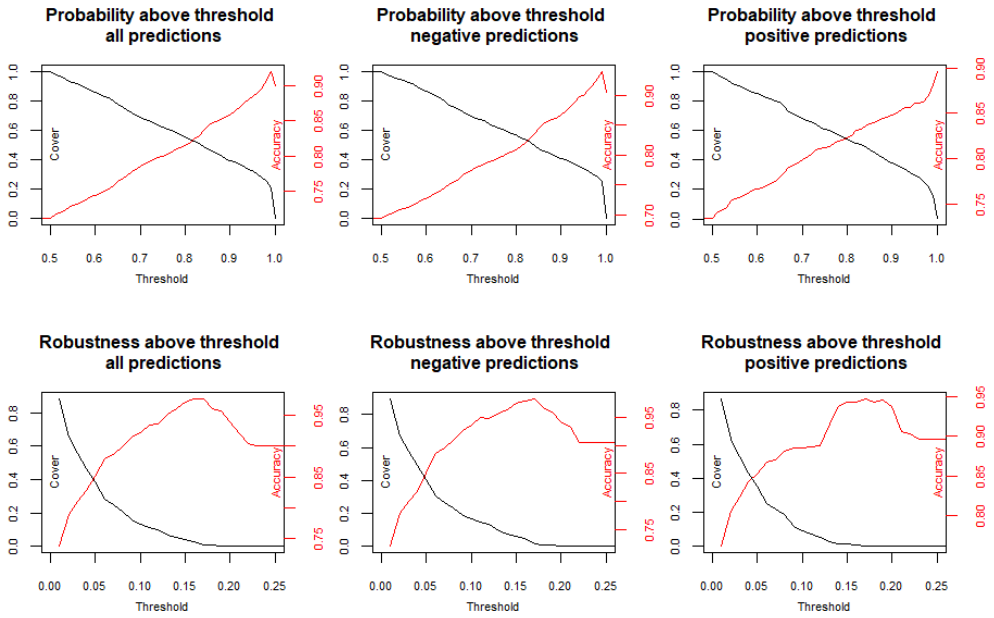


Figure 1: Graphs show cover, that is, the percentage of the cases that were classified if one only classifies the cases with measure (either probability or robustness value) above the given threshold, and accuracy over those cases. The final part of the curves is non-monotonic likely due to small sample size (very low cover).

classification as the most imprecise CSPN (based on the ε-contamination of the weights) for which a single class is non-dominated (one could also define some contamination for the continuous leaves, for instance by placing an interval of length ε around the *precisely* learned mean of the Gaussian distributions). The overall procedure runs a bisection with ε-contaminated CSPNs until converging to the (numerically approximate) maximum ε such that the prediction from the model is still unique (that is, all SPNs represented by the CSPN yield the same class prediction).

3. Case Study

Currently all orders placed at a major online retailer are evaluated through sophisticated hand-crafted business rules. The business rules can result in three outcomes: approval, outright cancellation, and manual review. A team of business analysts works around the clock to approve or cancel the orders that were flagged for manual review. For this case study, 36707 orders were collected and analysed, each of which was flagged for manual review by the business rules. For each of these orders, we collected the variables utilised by the existing expert system, the payment data for those orders, and the customer support data. This effort resulted in a total of 109 features. From those orders, the business analysts approved 18739 (51%), while 17968 (49%) were determined to be fraudulent and subsequently cancelled. The orders were labelled as follows: *true positive*: cancelled by analyst without customer complaint; *false positive*: cancelled by analyst, but the customer contacted customer support with a reasonable explanation; *true negative*: approved and subsequently paid for; *false negative*: approved, but not paid for (incurring loss to the company). The analysts achieved an accuracy of approximately 94% in this dataset. It should be noted, however, that the true accuracy could be lower. Not all customers might contact customer support upon cancellation of their order. Some might opt to forego their order, or simply order from a competitor.

We selected the 24 most important features to build an SPN: one continuous (that is, the price) and 23 Boolean variables. The SPN was learned using the procedure of [4], with the exception that independence tests are performed using one of Chi-square, Kruskal-Wallis or Kendall (according to the variables involved), clustering is done with the Gower distance (so as to take into account both categorical and

continuous variables), and leaves related to continuous variables are forced to be normally distributed (in this study, we have used a single continous variable). Then the robustness value is calculated per testing instance using the same approach as in [6]. Both robustness and probability of most probable class are used in order to discriminate the quality of the predictions. Figure 1 shows the results obtained by issuing a classification only when the model output is deemed *robust*, that is, either the probability value of the SPN (first row of graphs) or the robustness value from the CSPN (second row of graphs) for that particular instance was above a threshold (all possible thresholds are plotted). This is equivalent to saying that we refrain from guessing for those cases of greater indecisiveness. Based only on probabilities, no value of threshold would lead to classification results as accurate as the business analyst. Note that the analyst does not know which instances are robust or not, so they need to predict all of them. On the other hand, using robustness from CSPNs, if we only issued a decision when robustness is above the threshold of 0.1, then the model achieves the same performance as the analyst and would cover (that is, issue predictions for) about 15% of all orders. This can potentially benefit the company by reducing the time required to analyse orders flagged for review. This is a promising preliminary study of CSPNs with continuous and categorical variables. Such capability extends the applicability of CSPNs to many new domains.

References

[1] D. Conaty, D. D. Mauá, and C. P. de Campos. Approximation complexity of maximum a posteriori inference in sum-product networks. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence*, pages 322–331, 2017.

[2] D. Conaty, J. Martinez del Rincon, and C. P. de Campos. Cascading sum-product networks using robustness. In *Proceedings of Machine Learning Research 72*, pages 73–84, 2018.

[3] A. Darwiche and G. M. Provan. Query DAGs: A practical paradigm for implementing belief-network inference. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence*, pages 203–210, 1996.

[4] R. Gens and P. Domingos. Learning the structure of sum-product networks. In *Proceedings of the Thirtieth International Conference on Machine Learning*, pages 873–880, 2013.

[5] D. D. Mauá, D. Conaty, F. G. Cozman, K. Poppenhaeger, and C. P. de Campos. Robustifying sum-product networks. *International Journal of Approximate Reasoning*, 101:163–180, 2018.

[6] D. D. Mauá, F. G. Cozman, D. Conaty, and C. P. de Campos. Credal sum-product networks. In *Proceedings of the Tenth International Symposium on Imprecise Probability: Theories and Applications*, pages 205–216, 2017.

[7] A. Nath and P. Domingos. Learning tractable probabilistic models for fault localization. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1294–1301, 2016.

[8] R. Peharz, R. Gens, F. Pernkopf, and P. Domingos. On the latent variable interpretation in sum-product networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–14, 2016.

[9] H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 337–346, 2011.

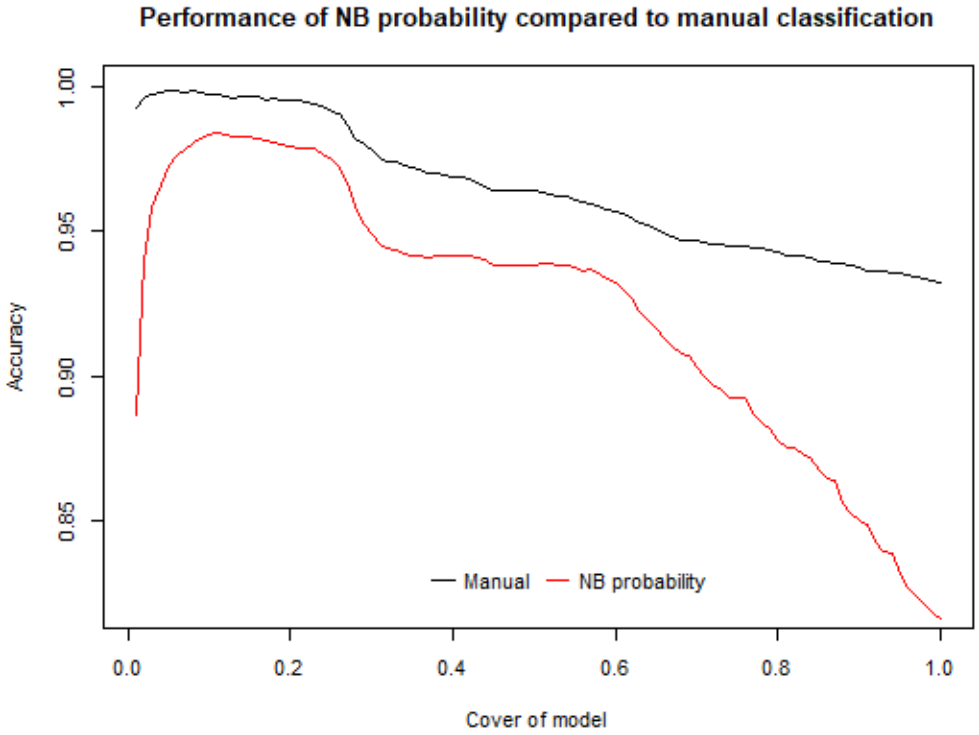
[10] F. Rathke, M. Desana, and C. Schnörr. Locally adaptive probabilistic models for global segmentation of pathological oct scans. In *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention*, pages 177–184, 2017.

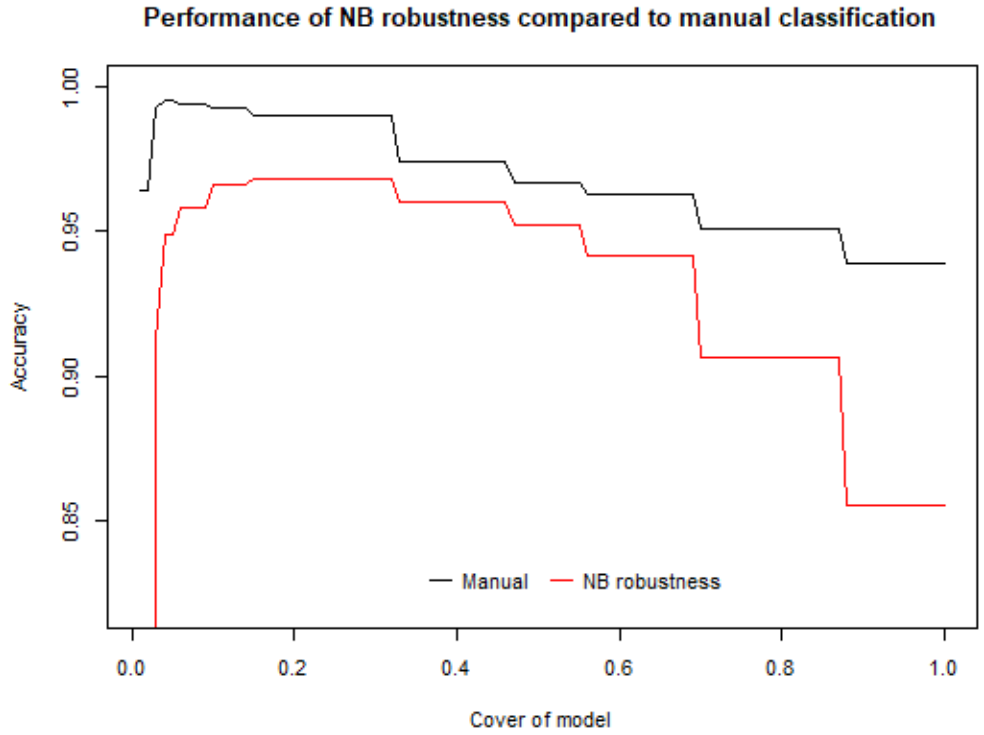
[11] H. Zhao, M. Melibari, and P. Poupart. On the relationship between sum-product networks and Bayesian networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 116–124, 2015.

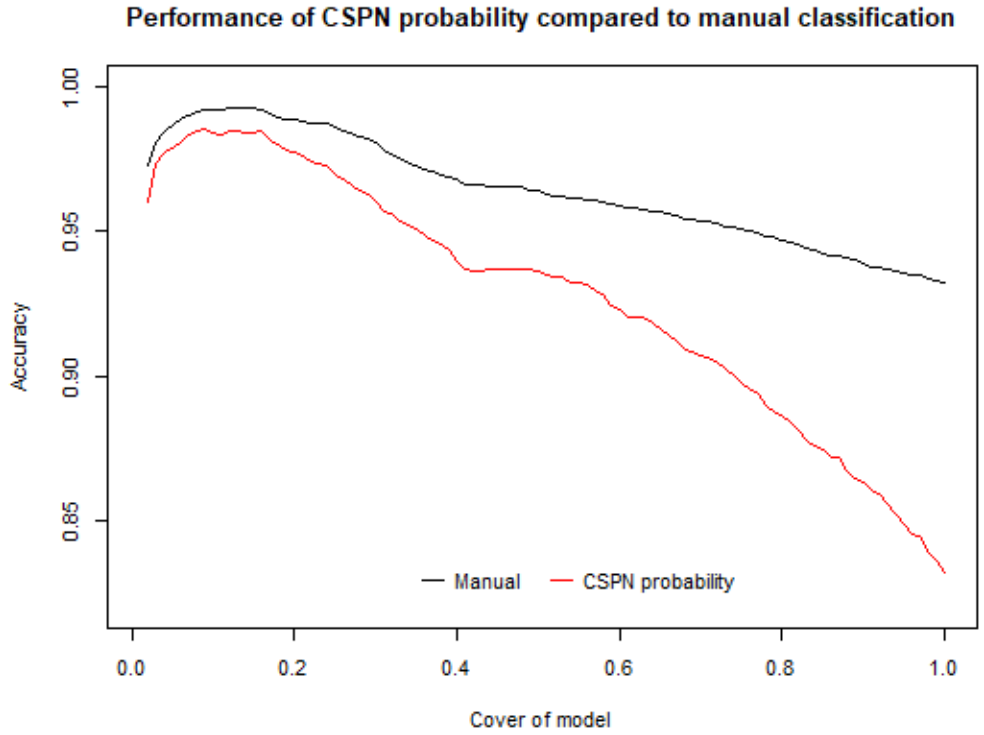
Appendix G

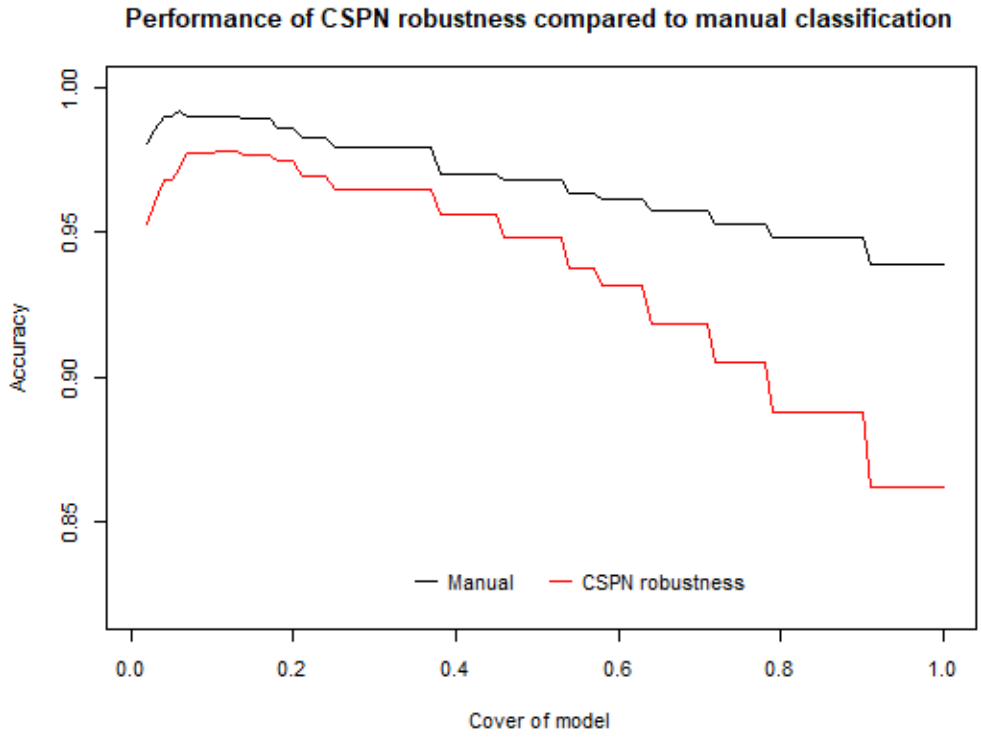
Classifiers' performance

The following visualisations show the accuracy of the NB and CSPN classifiers compared against the performance of manual review over the exact same instances. The visualisations are provided for subsets of the predictions based upon both of the classifiers reliability metrics: probability and robustness.




Performance of NB robustness compared to manual classification
Accuracy
1.00
0.95
0.90
0.85
Manual
NB robustness
0.0
0.2
0.4
0.6
0.8
1.0
Cover of model


Performance of CSPN probability compared to manual classification
Accuracy
1.00
0.95
0.90
0.85
Manual
CSPN probability
0.0
0.2
0.4
0.6
0.8
1.0
Cover of model


Performance of CSPN robustness compared to manual classification
Accuracy
1.00
0.95
0.90
0.85
Manual
CSPN robustness
0.0
0.2
0.4
0.6
0.8
1.0
Cover of model

Appendix H

R code model comparison

```
# Import models' results
rawNB <- read.csv2(file.choose())
rawSPN <- read.csv2(file.choose())
rawXGB <- read.csv2(file.choose())
rawMAN <- read.csv2(file.choose())

# Remove rownames
rawNB[,1] <- NULL
rawSPN[,1] <- NULL
rawXGB[,1] <- NULL
rawMAN[,1] <- NULL

# Create template for processing
resMAN <- data.frame(matrix(ncol = 5, nrow = nrow(rawMAN)))
colnames(resMAN) <- c("Actual", "Predicted", "Correct", "Prob
    ↪ ", "Rob")
resMAN$Actual <- rawMAN[, 1] > 0

# Copy layout for other models
resXGB <- resMAN
resSPN <- resMAN
resNB <- resMAN

resNB$Predicted <- rawNB[, 2] > 1
resNB$Correct <- resNB[, 1] == resNB[, 2]
resNB$Prob <- rawNB[, 3]
resNB$Rob <- rawNB[, 4]

resSPN$Predicted <- rawSPN[, 2] > 1
resSPN$Correct <- rawSPN[, 1] == rawSPN[, 2]
resSPN$Prob <- rawSPN[, 3]
resSPN$Rob <- rawSPN[, 4]

resMAN$Predicted <- rawMAN$VERDICT_ANALYST == 1
resMAN$Correct <- rawMAN$VERDICT_ANALYST == rawMAN$IS_FRAUD

resXGB$Predicted <- rawXGB >= 0.5
resXGB$Correct <- resXGB[, 1] == resXGB[, 2]
resXGB$Prob <- rawXGB[, 1]
```

```
resXGB[which(resXGB$Prob < 0.5), "Prob"] <- 1- resXGB[which(
    ↪ resXGB$Prob < 0.5), "Prob"]

# Convert to numerics
resNB$Prob <- as.numeric(as.character(resNB$Prob))
resNB$Rob <- as.numeric(as.character(resNB$Rob))
resSPN$Prob <- as.numeric(as.character(resSPN$Prob))
resSPN$Rob <- as.numeric(as.character(resSPN$Rob))
resXGB$Prob <- as.numeric(as.character(resXGB$Prob))


# Create confusion matrices
confusionMatrices <- data.frame(matrix(ncol = 6, nrow = 4))
colnames(confusionMatrices) <- c("Model", "Accuracy", "TN", "
    ↪ FN", "FP", "TP")

getConfusion <- function(name, results){
  length <- nrow(results)
  TP <- (nrow(results[results$Actual == TRUE & results$
      ↪ Predicted == TRUE, ])) / length
  FP <- (nrow(results[results$Actual == FALSE & results$
      ↪ Predicted == TRUE, ])) / length
  TN <- (nrow(results[results$Actual == FALSE & results$
      ↪ Predicted == FALSE, ])) / length
  FN <- (nrow(results[results$Actual == TRUE & results$
      ↪ Predicted == FALSE, ])) / length
  Accuracy <- (TP + TN)

  return(c(name, Accuracy, TN, FN, FP, TP))
}

confusionMatrices[1, ] <- getConfusion("Manual", resMAN)
confusionMatrices[2, ] <- getConfusion("XGBoost", resXGB)
confusionMatrices[3, ] <- getConfusion("SPN", resSPN)
confusionMatrices[4, ] <- getConfusion("NB", resNB)

# Calculate accuracy by cover
GetAccuracyByCover <- function(data, measure, cover){
  # Measure should be 4 for probability and 5 for robustness
  temp <- data[which(data[, measure] > quantile(data[,
      ↪ measure], 1 - cover)), ]
  return(mean(temp$Actual == temp$Predicted))
}

levelOfDetail <- 100
accuraciesByCover <- data.frame(matrix(ncol = 7, nrow =
    ↪ levelOfDetail))
colnames(accuraciesByCover) <- c("Cover", "Manual", "NB-prob"
    ↪ , "NB-rob", "SPN-prob", "SPN-rob", "XGB-prob")

for(i in 1:levelOfDetail){
```

```
  cover <- i / levelOfDetail
  accuraciesByCover[i, "Cover"] <- cover
  accuraciesByCover[i, "Manual"] <- confusionMatrices[1, "
    ↪ Accuracy"]
  accuraciesByCover[i, "NB-prob"] <- GetAccuracyByCover(resNB
    ↪ , 4, cover)
  accuraciesByCover[i, "NB-rob"] <- GetAccuracyByCover(resNB,
    ↪  5, cover)
  accuraciesByCover[i, "SPN-prob"] <- GetAccuracyByCover(
    ↪ resSPN, 4, cover)
  accuraciesByCover[i, "SPN-rob"] <- GetAccuracyByCover(
    ↪ resSPN, 5, cover)
  accuraciesByCover[i, "XGB-prob"] <- GetAccuracyByCover(
    ↪ resXGB, 4, cover)
}

accuraciesByCover[accuraciesByCover == "NaN"] <- NA


# Calculate manual performance given another measure
manualPerformanceByModel <- function(data, measure = 4, lod =
  ↪  100){
  df <- data.frame(matrix(ncol = 4, nrow = lod))
  colnames(df) <- c("Cover", "ModelAccuracy", "ManualAccuracy
    ↪ ", "CombinedAccuracy")

  for(i in 1:lod){
    cover <- i / lod

    tempModel <- data[which(data[, measure] > quantile(data[,
      ↪  measure], 1 - cover)), ]
    tempManual <- resMAN[-which(data[, measure] > quantile(
      ↪ data[, measure], 1 - cover)), ]

    accuracyModel <- mean(tempModel$Actual == tempModel$
      ↪ Predicted)
    accuracyManual <- mean(tempManual$Actual == tempManual$
      ↪ Predicted)

    accuracyCombined <- (accuracyModel * cover) + (
      ↪ accuracyManual * (1 - cover))

    df[i, 1] <- cover
    df[i, 2] <- accuracyModel
    df[i, 3] <- accuracyManual
    df[i, 4] <- accuracyCombined
  }

  df[lod, 3] <- NA
  return(df)
}
```

```
impactManualFromXGB <- manualPerformanceByModel(resXGB,
    ↪ measure = 4, lod = 100)
impactManualFromNBProb <- manualPerformanceByModel(resNB,
    ↪ measure = 4, lod = 100)
impactManualFromNBRob <- manualPerformanceByModel(resNB,
    ↪ measure = 5, lod = 100)
impactManualFromSPNProb <- manualPerformanceByModel(resSPN,
    ↪ measure = 4, lod = 100)
impactManualFromSPNRob <- manualPerformanceByModel(resSPN,
    ↪ measure = 5, lod = 100)



# Calculate manual accuracies over the same instances as the
    ↪ models
manualPerformanceOverSameInstances <- function(data, measure
    ↪ = 4, lod = 100){
  df <- data.frame(matrix(ncol = 3, nrow = lod))
  colnames(df) <- c("Cover", "ModelAccuracy", "ManualAccuracy
      ↪ ")

  for(i in 1:lod){
    cover <- i / lod

    tempModel <- data[which(data[, measure] > quantile(data[,
        ↪  measure], 1 - cover)), ]
    tempManual <- resMAN[which(data[, measure] > quantile(
        ↪ data[, measure], 1 - cover)), ]

    accuracyModel <- mean(tempModel$Actual == tempModel$
        ↪ Predicted)
    accuracyManual <- mean(tempManual$Actual == tempManual$
        ↪ Predicted)

    df[i, 1] <- cover
    df[i, 2] <- accuracyModel
    df[i, 3] <- accuracyManual
  }

  return(df)
}

sameCoverManualAndXGB <- manualPerformanceOverSameInstances(
    ↪ resXGB, 4, 100)
sameCoverManualAndNBProb <-
    ↪ manualPerformanceOverSameInstances(resNB, 4, 100)
sameCoverManualAndNBRob <- manualPerformanceOverSameInstances
    ↪ (resNB, 5, 100)
sameCoverManualAndSPNProb<-
    ↪ manualPerformanceOverSameInstances(resSPN, 4, 100)
```

```
sameCoverManualAndSPNRob <-
    ↪ manualPerformanceOverSameInstances(resSPN, 5, 100)



# Get XGB results with varying levels of cutoff
resXGBVaried <- as.data.frame(matrix(data = NA, nrow = 100,
    ↪ ncol = 11))
```

Appendix I

R code model comparison visualisations

```
library(ggpubr)

colMAN <- "black"
colXGB <- "red"
colNB <- "forestgreen"
colSPN <- "blue"
linewidth <- 1

# Plot performance of various metrics
plot.new()
par(mar=c(4,4,4,4), cex = 0.7)
plot(main = "Performance of reliability metrics",
    ↪ accuraciesByCover$Cover, accuraciesByCover$Manual, type
    ↪  = "l", xlab = "Cover", ylab = "Accuracy", ylim = c(.8,
    ↪  1))
lines(accuraciesByCover$Cover, accuraciesByCover$`XGB-prob`,
    ↪ col = colXGB, type = "l", lty = "solid", lwd =
    ↪ linewidth)
lines(accuraciesByCover$Cover, accuraciesByCover$`NB-prob`,
    ↪ col = colNB, type = "l", lty = "solid", lwd = linewidth
    ↪ )
lines(accuraciesByCover$Cover, accuraciesByCover$`NB-rob`,
    ↪ col = colNB, type = "l", lty = "dotted", lwd =
    ↪ linewidth)
lines(accuraciesByCover$Cover, accuraciesByCover$`SPN-prob`,
    ↪ col = colSPN, type = "l", lty = "solid", lwd =
    ↪ linewidth)
lines(accuraciesByCover$Cover, accuraciesByCover$`SPN-rob`,
    ↪ col = colSPN, type = "l", lty = "dotted", lwd =
    ↪ linewidth)

legend(x = "bottom", inset = .05,
       bty = "n",
       ncol = 3,
```

```
        legend = c("Manual", "XGB probability", "NB
            ↪ probability", "NB robustness", "CSPN probability
            ↪ ", "CSPN robustness"),
        col = c(colMAN, colXGB, colNB, colNB, colSPN, colSPN),
        lty = c("solid", "solid", "solid", "dotted", "solid",
            ↪ "dotted"),
        lwd = 1
        )

# Plot impact of using the models/metrics on manual
    ↪ performance
plotImpactOnManual <- function(data, metric = "NA", col = "
    ↪ red"){
        title <- paste("Performance of", metric, "and its
            ↪ impact on manual accuracy")

        plot.new()
        par(mar=c(5,4,4,5) + 0.1)
        plot(main = title, data[, 1], data[, 2], type = "l",
            ↪ xlab = "Cover of model", ylab = "Accuracy",
            ↪ ylim = c(.8, 1), col = col)
        lines(data[, 1], data[, 3], col = "black", type = "l"
            ↪ , lty = "solid")

        legend(x = "bottom", inset = .05,
               bty = "n",
               ncol = 2,
               legend = c("Manual", metric),
               col = c("black", col),
               lty = c("solid", "solid"),
               lwd = 3
        )
}

# Plot performance of combined approaches:
# Overall accuracies if model handles X and analyst handles Y
plot.new()
par(mar=c(5,4,4,5) + 0.1)
plot(main = "Performance of combined approach: model and
    ↪ manual", impactManualFromXGB$Cover, impactManualFromXGB
    ↪ $CombinedAccuracy, type = "l", col = colXGB, xlab = "
    ↪ Cover of model", ylab = "Accuracy", ylim = c(.8, 1))
lines(impactManualFromNBProb$Cover, impactManualFromNBProb$
    ↪ CombinedAccuracy, col = colNB, type = "l", lty = "solid
    ↪ ", lwd = linewidth)
lines(impactManualFromNBRob$Cover, impactManualFromNBRob$
    ↪ CombinedAccuracy, col = colNB, type = "l", lty = "
    ↪ dotted", lwd = linewidth)
lines(impactManualFromSPNProb$Cover, impactManualFromSPNProb$
    ↪ CombinedAccuracy, col = colSPN, type = "l", lty = "
    ↪ solid", lwd = linewidth)
```

```
lines(impactManualFromSPNRob$Cover, impactManualFromSPNRob$
    ↪ CombinedAccuracy, col = colSPN, type = "l", lty = "
    ↪ dotted", lwd = linewidth)

legend(x = "bottom", inset = .05,
       bty = "n",
       ncol = 3,
       legend = c("XGB probability", "NB probability", "NB
          ↪ robustness", "CSPN probability", "CSPN
          ↪ robustness"),
       col = c(colXGB, colNB, colNB, colSPN, colSPN),
       lty = c("solid", "solid", "dotted", "solid", "dotted")
          ↪ ,
       lwd = 3
)


# Plot performance of model and manual over same cases
plot.new()
par(mar=c(4,4,4,4), cex = 0.7)
plot(main = "Accuracy of manual and XGBoost probability over
    ↪ same cases", sameCoverManualAndXGB[, 1],
    ↪ sameCoverManualAndXGB[, 2], type = "l", xlab = "Cover",
    ↪  ylab = "Accuracy", ylim = c(.8, 1), col = "red")
lines(sameCoverManualAndXGB[, 1], sameCoverManualAndXGB[, 3],
    ↪  col = "black", type = "l", lty = "solid")

plotDifferencePerformance <- function(data, metric = "NA",
    ↪ col = "red"){
        title <- paste("Performance of", metric, "compared to
           ↪ manual classification")

        plot.new()
        par(mar=c(4,4,4,4), cex = 0.7)
        plot(main = title, data[, 1], data[, 2], type = "l",
           ↪ xlab = "Cover of model", ylab = "Accuracy",
           ↪ ylim = c(.82, 1), col = col)
        lines(data[, 1], data[, 3], col = "black", type = "l"
           ↪ , lty = "solid")

        legend(x = "bottom", inset = .05,
               bty = "n",
               ncol = 2,
               legend = c("Manual", metric),
               col = c("black", col),
               lty = c("solid", "solid"),
               lwd = 1
        )
}
```

```
plotDifferencePerformance(sameCoverManualAndXGB, "XGB␣
    ↪ probability")
plotDifferencePerformance(sameCoverManualAndNBProb, "NB␣
    ↪ probability")
plotDifferencePerformance(sameCoverManualAndNBRob, "NB␣
    ↪ robustness")
plotDifferencePerformance(sameCoverManualAndSPNProb, "CSPN␣
    ↪ probability")
plotDifferencePerformance(sameCoverManualAndSPNRob, "CSPN␣
    ↪ robustness")
```