JAN GERARD KLEIN

Molding the Symbiosis between Human and Machine

Contributions to Anomaly Detection, Model Evaluation, and Active Learning

CENTRUM WISKUNDE & INFORMATICA | VRIJE UNIVERSITEIT AMSTERDAM | MINISTERIE VAN BINNENLANDSE ZAKEN & KONINKRIJKSRELATIES

Molding the Symbiosis between Human and Machine

Contributions to Anomaly Detection, Model Evaluation, and Active Learning

ISBN: 978-94-6421-780-3

Typeset by LaTeX.
Printed by: Ipskamp, proefschriften.net
Cover design by: Jolanda van Beusichem, www.pixapanda.com

VRIJE UNIVERSITEIT

Molding the Symbiosis between Human and Machine

Contributions to Anomaly Detection, Model Evaluation, and Active Learning

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. J.J.G. Geurts,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de Faculteit der Bètawetenschappen
op woensdag 7 september 2022 om 15.45 uur
in een bijeenkomst van de universiteit,
De Boelelaan 1105

door

Jan Gerard Klein

geboren te Den Helder

promotoren:	prof.dr. R.D. van der Mei
	prof.dr. S. Bhulai
copromotor:	prof.dr. M. Hoogendoorn
promotiecommissie:	prof.dr. G.M. Koole
	prof.dr. S.I. Birbil
	prof.dr. A. Nowé
	dr.mr. C.S. Gerritsen
	dr.ir. R.H.A. Lindelauf

Voorwoord

Het is hem gelukt, maar wat hij er uiteindelijk mee bereikt...
Dat is voor ons allemaal een groot raadsel.

Renate Verbaan
Wie is de Mol? 2007

Dit proefschrift is het resultaat van een interessant, leuk, soms frustrerend en iets langer dan verwacht promotietraject. Uiteindelijk is er sinds de start van mijn PhD-onderzoek bij het Centrum Wiskunde & Informatica (CWI), het Ministerie van Binnenlandse Zaken en Koninkrijksrelaties (MinBZK) en de Vrije Universiteit Amsterdam (VU) ruim vijf jaar verstreken voor ik aan dit voorwoord ben begonnen. Grappig eigenlijk dat hetgeen je als laatste hebt geschreven, als eerste wordt gelezen. Ik hoop dat je ook verder zult kijken in mijn proefschrift, want er staat best mooi onderzoek in wat mij betreft.

Ik zeg wel dat dit *mijn* proefschrift is, maar het onderzoek dat ik heb mogen uitvoeren had ik nooit alleen kunnen doen en is door veel mensen mogelijk gemaakt. Sommigen zijn hier direct bij betrokken geweest door mee te denken of gezamenlijk onderzoek uit te voeren, terwijl anderen me onbewust motiveerden tijdens niet-werkmomenten zoals koffiepauzes, borrels, dagjes uit en vakanties.

Allereerst wil ik mijn promotoren Rob en Sandjai en copromotor Mark bedanken voor hun enthousiasme, laagdrempeligheid en vele goede ideeën. Het wekelijks overleg dat we met elkaar hadden, was altijd gezellig en het was eerder regel dan uitzondering dat we over de tijd heen gingen. Voordat ik aan dit traject begon, had ik nooit verwacht dat ik met Mark zou belanden op Mysteryland en Amsterdam Dance Event of dat Rob en Sandjai mij voorbij zouden scheuren op een kartbaan. Ook was het fijn dat jullie lieten merken vertrouwen in mij te hebben door mij veel vrijheid te geven. Gelukkig stuurden jullie wel bij als het nodig was en stelden jullie belangrijke en kritische vragen.

Verder wil ik de leden van de promotiecommissie bedanken voor de tijd en moeite die zij gestoken hebben in het beoordelen van dit proefschrift en voor de discussiepunten die zij zullen opbrengen tijdens de verdediging.

Ook wil ik de betrokkenen bij MinBZK bedanken voor het in goede banen leiden van dit promotietraject. Dat heeft mij heel wat administratieve zorgen bespaard. Ik werd vrijgelaten om te onderzoeken wat ik zelf wilde, omdat er in het algemeen vertrouwen was in een goede uitkomst. Als ik aangaf iets nodig te hebben, dan werd meteen de procedure in gang gezet om het te regelen. Bovendien heb ik veel gehad aan de kennis binnen het ministerie, en dan met name op praktisch vlak. Mijn achtergrond is voornamelijk 'theoretisch', dus ik heb daar veel kunnen leren. Ook wil ik jullie bedanken voor de postdocpositie die jullie mij hebben aangeboden zodat ik mijn onderzoek voort kon zetten.

Projectgenoten Etienne en Joris wil ik bedanken voor de geweldige werksfeer en voor alle bruikbare feedback die jullie mij gegeven hebben. Ik heb jullie niet voor niets gevraagd om mijn paranimfen te zijn en gelukkig willen jullie die rol vervullen. In de ruim drie jaar dat we elkaar kennen, zijn we goede vrienden geworden. Waar we vooral goed in zijn, is discussiëren, maar altijd met respect voor de mening van de ander. Tevens komt onze humor overeen en zijn jullie er ook voor mij als het minder goed gaat. De behoefte aan discussies heeft er wel voor gezorgd dat we iets langer dan gepland over ons gezamenlijke project (Hoofdstuk 5) hebben gedaan, maar het uitwerken van een gouden idee kost nou eenmaal tijd. Het is gelukkig geen 'Jans muntje' geworden. Lang vormden Etienne, Joris en ik een groep met z'n drieën, maar in de loop van de tijd zijn mede-Songfestivalfans Arwin en Britt als waardevolle aanvullingen aan ons MinBZK-team toegevoegd. Wat mij betreft zijn we samen een gezellige groep die interessant onderzoek uitvoert dat ondanks mijn hartverscheurende vertrek hopelijk glorieus wordt voortgezet.

Als ik over gezelligheid en onderzoek praat, dan kan ik de Stochasticsgroep van het CWI onder leiding van Bert Zwart niet zomaar vergeten om te bedanken, zo ook niet de drie onderzoeksgroepen van de VU waar ik onderdeel van ben geweest: Analytics & Optimization (A&O) o.l.v. Ger Koole, Computational Intelligence (CI) o.l.v. Guszti Eiben en Quantitative Data Analytics (QDA) o.l.v. Mark. Ik heb in die ruim vijf jaar tijd zoveel mensen leren kennen die ik persoonlijk wil bedanken voor hun gezelligheid en onderzoeksideeën, maar als ik dat doe dan wordt dit voorwoord langer dan elk ander hoofdstuk (en ik heb al genoeg tijd aan dit proefschrift besteed). Daarom noem ik een aantal mensen specifiek. Van het CWI zijn dit Guus, Robin en Rebekka. Guus en Robin waren de buren van mijn kamergenoot Joris en van mij. Hoe vaak zijn we wel niet bij elkaar binnengelopen als we hoorden dat er een discussie gaande was? Rebekka verdient een shout-out omdat zij de vrolijke noot van de groep is met wie ik veel kan lachen (en met wie ik kan klagen over van alles). Van de A&O-groep wil ik Yura en Ανδρονικη (Anni) noemen met wie ik een goede band heb gekregen. Het vieren van de jaarwisseling bij Yura blijft een hoogtepunt (de gevolgen door de Moscow mules waren iets minder). The walks that Anni and

I started in Vondelpark during the second lockdown have become a tradition and hopefully will stay like that (without COVID though!). I really appreciate that you are so enthusiastic (especially your laugh) and that I can talk with you about anything. Also, I would like to thank the ‘book people’ from the A&O group for introducing me to new book genres and inviting me to their homes for food and talks. Hiervan wil ik Bernard en Corné speciaal noemen met wie ik gelijktijdig aan het promotietraject ben begonnen en met wie ik veel samen heb gewerkt voor PhD-vakken. Zonder jullie was het een stuk lastiger geweest om die vakken met een mooi cijfer af te ronden. Furthermore, I would like to thank Alessandro, Ali, Floris, David, and Luis who made the CI/QDA Tuesday a nice work day and who played essential roles at De Tegenstelling and De Blauwe Engel.

Naast de vele mede-promovendi die ik goed heb leren kennen, wil ik ook een aantal vrienden die ik daar buiten al kende bedanken. Allereerst Caroline. Toen we elkaar tijdens de master leerden kennen, raakten we al snel goed bevriend (na een ietwat stroef begin omdat je dacht dat ik afstandelijk was). We hebben veel samengewerkt, maar voornamelijk heel veel gepraat. Vooral je openheid en spontaniteit waardeer ik. Dat zijn twee eigenschappen waar ik zelf beter in kan worden. Ten tweede wil ik Daniël bedanken naar wie ik al jarenlang autobiografische epistels mag sturen (en natuurlijk stuur jij die ook naar mij). Ten derde wil ik mijn ‘Molvrienden’ Mathieu en Jolanda bedanken die mij zo ontzettend aan het lachen hebben gemaakt en dat nog steeds doen. Of we nou een avondje gamen terwijl we bellen, of een uitstapje naar een Nederlandse of Belgische stad doen. Ik wil hierbij ook de andere mensen die ik dankzij *Wie is de Mol?* heb leren kennen bedanken voor de leuke gesprekken en activiteiten. Met name dank aan Daan voor het zijn van een Molencyclopedie en het consequent onthouden van leuke quotes zodat ik de meest toepasselijke in dit proefschrift heb kunnen zetten. Ik wil Jolanda nog eens extra bedanken voor het ontwerpen van de kaft en de boekenlegger van dit proefschrift en voor het maken van de afbeeldingen die de drie thema’s inluiden. Alles ziet er werkelijk prachtig uit.

Voordat ik afsluit, wil ik Opa Cornelissen en Tante Gerrie noemen die helaas niet meer bij ons zijn. Jullie waren altijd geïnteresseerd in mij en daarom vind ik het zo jammer dat ik dit proefschrift niet meer aan jullie kan geven.

Als laatste wil ik mijn familie en andere vrienden en kennissen bedanken voor de steun, feedback en leuke momenten die ze mij gegeven hebben. Bijvoorbeeld Oom Thijs en Tante Riet bij wie ik vaak heb mogen aanschuiven om te genieten van een heerlijk diner dat zo in een sterrenrestaurant opgediend had kunnen worden. Speciaal wil ik ook mijn ouders Gerard en Lida, zus Marije, zwager Mohammad en nichtjes Sophia, Elina en Alisa noemen. Jullie staan altijd voor mij klaar en stimuleren mij om te doen wat ik leuk vind. Het is fijn om zo’n betrouwbare en lieve groep mensen te hebben om op terug te vallen.

Contents

Nomenclature

Abbreviations

F_β	F_β score
AD	Anomaly Detection
AL	Active Learning
ASeSL	Active Semi-Supervised Learning
FN	(Number of) False Negative(s)
FP	(Number of) False Positive(s)
GBM	Gradient Boosting Machine
IF	Isolation Forest
ME	Model Evaluation
ML	Machine Learning
NID	Network Intrusion Detection
NIDS	Network Intrusion Detection System
PUL	Positive-Unlabelled Learning
RQ	Research Question

SeSL	Semi-Supervised Learning
SL	Supervised Learning
TN	(Number of) True Negative(s)
TP	(Number of) True Positive(s)
UL	Unsupervised Learning

Mathematical notation

δ_a^β	Anomaly information metric
δ_z^β	Uncertainty information metric
Δ^γ	Dynamic update factor
$\hat{y}$	predicted output target value / predicted response value / predicted label
$\mathcal{L}$	Set of labelled observation indices
$\mathcal{M}$	Set of observation indices
$\mathcal{Q}$	Set of query observation indices
$\mathcal{U}$	Set of unlabelled observation indices
L	Number of labelled observations
Q	Number of query observations
U	Number of unlabelled observations

1 Introduction

Je hebt geen idee hoe spannend dit spel is.

Susan Visser
Wie is de Mol? 2014

Nowadays, more and more data is collected in an increasing number of companies and across many research domains. Especially since the internet became publicly available in the 1990s, the collection of data has exploded [54]. It grew even more with the advent of social media (starting from 2005) and the Internet of Things (IoT) (starting from 2015), which are physical objects such as security cameras and wearable medical devices that exchange data in a network. Besides volume, *data variety* has grown too. Much more unstructured or raw data, such as text and videos, is collected compared to structured data. Lee [54] argues that this is enabled by technological advancements, making it easier for people and organisations to produce information. For example, an aeroplane is estimated to generate between 5 and 8 terabytes of raw data per flight in 2026 [123]. Or, consider all the information that every person with a smartphone generates each day.

Yet, such data usually lacks a coherent structure, which complicates the extraction of information from it. Therefore, more extensive computational and human efforts are required to construct useful and workable datasets. As more information is collected, also the amount of unlabelled data increases. Labels identify raw data and provide context. Consider computer network data, which consists of a huge number of connections that describe how computers, or hosts, communicate with each other. The label of a connection could indicate whether it is benign or malicious (a cyber attack). However, it is impossible for a cyber expert to determine the label for each connection [136]. This leads to large amounts of partially labelled or completely unlabelled data; not only in *network intrusion detection* (NID), but in many more domains [35]. This complicates the interpretation of data and can also lead to privacy concerns, because sensitive information could be hidden in the data.

Machine Learning (ML) is accurately described by Mitchell [74] as "the study of computer algorithms that improve automatically through experience". An ML algorithm tries to discover patterns in data without explicitly being told what to look for. Moreover, ML can automatically improve decision-making by evaluating the consequences of the choices that the algorithm has made [44]. There are several types of ML techniques, but historically they are separated into three groups based on the feedback that the algorithm receives during learning or training [7]:

- *Supervised Learning* (SL)
- *Unsupervised Learning* (UL)
- *Reinforcement Learning* (RL).

Supervised Learning In SL, the data instances with *feature values* (inputs) and their corresponding *target values* (e.g., labels) are available at training time. The method attempts to learn the function that maps the input to the target output. This allows the method to make predictions about the target values of new data instances.

Unsupervised Learning In UL, the learner is expected to find patterns in the feature values without using any target outputs. Although these target values are not available, the method can still make predictions about a new data instance. It can discover that the new observation has characteristics similar to some group of training instances, or that it is completely different.

Reinforcement Learning RL makes decisions in a dynamic environment in order to achieve a certain goal, such as playing a game of chess. The future choices that the method makes are directly determined by the rewards it obtained from previous decisions.

Although the three ML categories encompass most of the algorithms, a fourth category called *Semi-Supervised Learning* (SeSL) is also identified [15].

Semi-Supervised Learning SeSL falls between SL and UL, because it assumes that some target values are available, but not all data is required to be labelled. SeSL uses the unlabelled part of the data to improve the performance of the method trained on labelled data.

The categorisation of SL, UL, and SeSL is closely linked to the availability of labels. This also means that even though labelled data contains more information than data without target values, partially labelled or unlabelled data is far from useless. UL methods made for *Anomaly Detection* (AD) can be applied on unlabelled data, because anomalies are generally defined as observations with significantly different feature values. In Section 1.1, we provide background information on anomalies and their detection, and explain what challenges arise in the field of AD. One of the broader challenges, not only for AD, is *Model Evaluation* (ME), i.e., assessing the prediction performance of learning methods. In Section 1.2, we construct valuable tools to address this challenge. Issues mostly arise when it is difficult to obtain fully labelled data. Therefore, we introduce *Active Learning* (AL) for efficient data labelling in Section 1.3. The topics introduced in these three sections all contribute to *molding the symbiosis between human and machine.*

1.1 Anomaly Detection

In this section, we define what an anomaly or outlier is, discuss what kind of anomalies exist, and in what domains these kinds commonly occur. Furthermore, we examine how the different learning methods (SL, UL, and SeSL) are used for AD and how usable datasets are constructed for this purpose. Also, we discuss what problems arise in outlier detection, particularly the issues with the unsupervised evaluation of discovering anomalies. Then, we transform these challenges into *research questions* (RQs). Their answers are our contributions to the field of AD.

1.1.1 Anomalies and Techniques

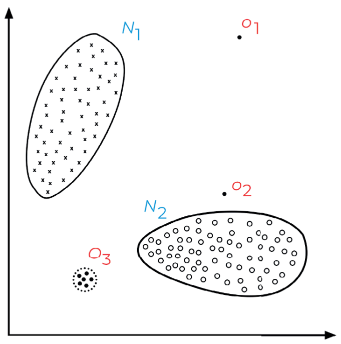


Figure 1.1: Example of anomalies in two-dimensional data (inspired by [14])

AD concerns itself with finding observations or patterns in data that are different than anticipated [14]. Most often, these instances are called *anomalies* or *outliers*. Figure 1.1 gives an illustration of anomalies in a dataset with only two features. It shows two distinct clusters N_1 and N_2 with observations that all have similar characteristics. There are two outliers o_1 and o_2, and one small possibly anomalous cluster O_3. It is important to identify such data points, as they could provide critical information.

AD is applied in many research fields such as intrusion detection, fraud detection, medical outlier detection, industrial damage detection, video surveillance, text anomaly detection, sensor networks, and IoT big data outlier detection (see the surveys in [13, 14]). It is crucial to find the anomalies in these areas, as outliers could indicate a dangerous or faulty process, such as a cyber attack, fraudulent activity, or system failure. Since there are so many application domains, there are also various kinds of anomalies that can be searched for. Also, each type of anomaly has different outlier detection techniques specialised for it. Three categories have been described [2, 13, 14]:

- point anomalies
- contextual anomalies
- collective anomalies.

Point Anomalies They refer to single observations that are considerably different from the rest. Take for example credit card data: when there is a very large transaction that is way higher than usual for the specific owner, then this observation is considered to be a point anomaly.

Contextual Anomalies This category refers to observations that are anomalous only in a specific context. Say rent is always deducted on the 27th of the month, then a rent payment on the 8th is a contextual anomaly. The deduction of rent is in itself not an outlier, but in this specific situation it is.

Collective Anomalies These observations are not necessarily anomalous by themselves, but collectively they are when compared to the rest of the data. If one's credit card is stolen, often the thief makes transactions of small amounts to not seem suspicious [81]. Each transaction on its own is not an outlier, but the set of these transactions is collectively anomalous.

Besides anomaly types, the choice of outlier detection techniques depends on the availability of data labels too [14]. In this binary setting, the label of an observation denotes whether it is normal or anomalous. Due to the different stages of label availability, supervised, unsupervised, and semi-supervised AD methods have been developed.

Supervised Anomaly Detection SL methods are designed for fully labelled data with normal and anomalous observations. Therefore, learning becomes a binary prediction problem. Gogoi et al. [33] state that the anomalies should have some *signature* for SL techniques. To this end, anomalous characteristics can be learned and similar unseen outliers can be detected. This immediately shows one of the limitations: new anomalies with different characteristics are not easily detected. Furthermore, it is often not feasible to label a sufficiently large dataset to capture the complex underlying structure. This limits the applicability of SL methods in practice.

Unsupervised Anomaly Detection UL methods do not require any labels at all, and as a result, have fewer practical limitations. The premise is to build a model of expected behaviour and then detect anomalies as observations that deviate from this behaviour [33]. Although UL techniques are much better in detecting novel anomalies than SL, they struggle to obtain an overall good detection rate [37].

Semi-Supervised Anomaly Detection SeSL techniques are in between SL and UL. The methodology is usually deduced from supervised algorithms, but with the added benefit of using unlabelled data [37]. For example, if a data point with characteristics that resemble an anomalous signature is located in a region with many unlabelled observations, then it is less likely to be classified as an anomaly than if it is located in a low-density region. This means SeSL methods combine the strengths of SL and UL, which could lead to better detection performance.

1.1.2 Challenges in Anomaly Detection

The research on AD and its application is widely spread, but there are common challenges. We discuss three important ones.

Lack of Decent Data It is often difficult to obtain labelled data that is both correct and representative for all types of behaviour. This is especially the case for anomalous observations. Ahmed et al. [2] and Sommer and Paxson [103] discuss this specifically for the field of NID, but they state that it is not limited to the cyber security domain. Moreover, much of the raw data is unstructured, for example it has variables whose entries are comments written by people. Or, there could be some missing values for certain features, completely missing variables, or multiple features that are highly similar. *Feature engineering* and *feature extraction* are necessary tools to convert data into a usable dataset. Feature engineering is the act of constructing new variables from existing features using human insights of the domain [135]. In computer network data, the relative difference between the number of bytes sent and received can provide valuable information. Hence, this feature is constructed from the two variables and added to the feature set. In feature extraction, variables that are redundant or non-informative are transformed or removed. When the data has two features that always add up to 1, then one of them is redundant and could therefore be removed.

Computational Complexity Chalapathy and Chawla [13] and Chandola et al. [14] discuss some other challenges in AD research, one of which is *computational complexity*. In an offline setting, extensive training can be done beforehand, and testing is usually done quickly. However, complications arise in an online setting when the model is retrained often. Outlier detection techniques commonly have an extensive training procedure (in terms of long learning times and large memory usage). Therefore, which technique to select is an important factor in constructing an AD method. The survey by Ahmed et al. [2] can aid in this, as it specifically lists the computational complexity of many AD methods.

Unsupervised Evaluation Finally, the evaluation of the general performance of a prediction model is a major issue in UL [48]. Because labels are not available, it is difficult to judge whether the method has predicted well. Faced by this, Goldstein and Uchida [34] propose a 'comparative evaluation' of many AD techniques. For each algorithm, they rank the resulting anomaly scores, apply different thresholds for predicting something as normal or anomalous, and construct the *receiver operator characteristic* (ROC) curve. The area under this curve quantifies the prediction performance. Note that it is again necessary to know some of the actual anomalous observations, i.e., to have some labels, which are difficult to come by. However, this information can be provided by cleverly incorporating a human domain expert.

1.1.3 Research Questions in Anomaly Detection

We rephrase the challenges described in the previous subsection in the following RQs:

1*a* *How can we make the raw data of two real case studies usable for AD?*

1*b* *What are the strengths and weaknesses of the SL and UL techniques used in these case studies?*

1*c* *What are the benefits of using a human domain expert to inspect the anomalous instances that the ML technique discovered?*

These three questions are dissected in Part I, which consists of Chapters 2 and 3. The answers are summarised and discussed in Chapter 8.

1.1.4 Our Contributions to Anomaly Detection

RQ 1*a* is explored in Chapters 2 and 3. In these chapters, we perform feature engineering and feature extraction on the raw data of two novel case studies to create usable datasets. The first case study is aimed at discovering malicious network connections in data that was captured during a realistic international cyber defence exercise. The second case study entails the identification of fraudulent bookings in datasets of aeroplane bookings made by online travel agencies. We use domain knowledge to engineer relevant new features, remove redundant variables, and combine specific features to make them more informative. We show that these variables are important in the discovery of outliers, and hence, they provide valuable insights to AD in cyber security and in booking fraud detection.

To address RQ 1*b*, we consider two levels of label availability, namely *(i)* only partial labels, and *(ii)* no labels. In Chapter 2, only some malicious observations have been labelled beforehand. It is undetermined whether other instances are malicious or not. We apply both unsupervised and supervised techniques to examine their strengths and weaknesses on network data. We do not need labels for UL, but we do use the known malicious observations to see how accurately they are found. In SL, we assume that the unlabelled observations are all normal (benign), because malicious connections are relatively rare. Under these unique assumptions, we demonstrate interesting differences between the UL and SL paradigms in NID. In Chapter 3, the data consists of online aeroplane bookings and is completely unlabelled. Hence, we are limited to using UL methods. Since several techniques are considered, we compare their results on booking data containing some fraudulent observations. Additionally, the two case studies both operate in an offline setting, meaning that potentially long training times are only encountered once. In Chapters 6 and 7, we have to work in an online setting. Therefore, we specifically take the computational complexity into account and also move towards SeSL techniques. We show that online approaches obtain favourable results without putting a strain on computational resources.

Finally, RQ 1*c* is answered by using human domain experts for an assessment of the results. In Chapters 2 and 3, we present them a set of anomalies that the detection models have found and ask the experts to indicate which observations are indeed malicious. Moreover, to evaluate how well benign observations are classified, we also show the experts non-anomalous data and ask them whether these instances are normal. This expert evaluation helps us to answer RQ 1*b* as well. Moreover, it sets our work apart from research that applies unsupervised techniques and only provides an intrinsic assessment of the results, e.g., indicating how many clusters there are, how dense they are, and so on.

1.2 Model Evaluation

In Section 1.1, we proposed including a human domain expert for the evaluation of unsupervised ML methods. In this section, we explain how an *evaluation metric* quantifies performance. Then, we take two different routes. First, we discuss what problems arise with Model Evaluation when we only have partially labelled data, and how we contribute to solving this problem. Second, we examine more fundamental issues that evaluation metrics have even for fully labelled data and how our research provides a framework that alleviates these underlying problems.

1.2.1 Supervised Binary Classification

In supervised binary classification, the target value, or label, is predicted using a function f that maps the $K \in \mathbb{N}_{>0}$ feature variables to either one of two classes: the *negative* class '0' or the *positive* class '1'. Generally, the dataset is split into a training set and a test or evaluation set. The latter is used to assess the prediction performance of the trained model [7]. The trained supervised model $f : \mathbb{R}^K \to \{0, 1\}$ outputs the predicted label of an observation given its feature values. All predictions in the evaluation set are compared with the true classes to see how much they correspond, and consequently, how well the model performs.

Base Measures An evaluation metric quantifies the prediction performance. Most metrics are defined as a function of one or more of the four *base measures*:

- *number of true positives* (TP)
- *number of false negatives* (FN)
- *number of true negatives* (TN)
- *number of false positives* (FP).

TP is the number of truly positive instances that are correctly predicted, while FN is the number of those that are incorrectly predicted. The same reasoning holds for TN and FP, but with truly negative instances.

Many evaluation metrics combine base measures to provide one bounded value that is used for overall evaluation. For example, consider the F_1 score, which combines three base measures:

$$F_1 = \frac{2\text{TP}}{2\text{TP} + \text{FN} + \text{FP}}. \tag{1.1}$$

A higher F_1 score means that the model performs better [17]. When the evaluation data is fully labelled, there are no complications in calculating this metric. However, in this dissertation we examine whether a good and robust equivalent to the F_1 score for partially labelled data can be constructed. Later on, we analyse evaluation metrics more fundamentally. For example, the F_1 score takes values in $[0, 1]$, but which values are considered bad and which good?

1.2.2 Evaluation Metrics for Partially Labelled Data

For most evaluation metrics it is necessary to know the true labels to determine the base measures. As Equation (1.1) shows, it is necessary for the F_1 score to know which observations are truly positive or negative in order to determine TP, FN, and FP. Therefore, we explore the field in which we have partially labelled data and where we cannot easily calculate the base measures. More specifically, we consider *Positive-Unlabelled Learning* (PUL). This domain consists mostly of SeSL methods that learn from data containing either unlabelled or positively labelled observations [22, 110, 134]. Hence, if an observation is labelled, then by definition it is positive. This means there are no negative instances known, making it impossible to calculate TN and FP. Moreover, not all positive observations are labelled, which complicates the computation of TP and FN too. Evaluation metrics or PUL strategies that can deal with these limitations have been proposed, with the measure by Lee and Liu [55] (the LL score) as a prominent example. This measure has approximately the same behaviour as the F_1 score without the need for negative labels.

Research Question In absolute value the LL score can be rather different from the F_1 score. Moreover, the LL score is not always the highest for the model that in reality achieves the best F_1 score. Therefore, we focus on the following RQ:

2*a* *Is it possible to construct an accurate and robust equivalent to the common F_1 score for partially labelled data?*

This question is at the centre of Chapter 4. An explicit answer is given and discussed in Chapter 8.

Contributions We show both theoretically and experimentally in Chapter 4 that our PUL estimator of the F_1 score works better than the commonly used LL score under mild assumptions. First, it is closer to the true F_1 score, which is calculated after making all labels available. Second, our estimator is better at selecting the optimal model out of a set of models than the LL score is.

1.2.3 Benchmarking Performance Scores

Up to now, we have mostly discussed problems with quantifying the performance of ML models when the data is not fully labelled. However, there are more fundamental issues in evaluation. We do not mean problems specific to some evaluation metric such as the F_1 score, but an issue that occurs during the assessment process in general. Say we have trained a supervised ML model and we evaluate it on test data by means of an evaluation metric. This yields some value that quantifies the prediction performance, but the question remains how to interpret this score. For example, the F_1 score is bounded in $[0, 1]$, so a score of 0.8 intuitively sounds good. But this can be deceptive: it is possible that it is rather easy to attain such a score on this specific evaluation set. It can also be the other way around, a score of 0.4 sounds bad, but can actually be very good.

Therefore, we need to have a point of reference, a *baseline*, to compare the evaluation score with. Usually, a state-of-the-art technique is used to compare a newly developed method with. However, the results are highly dependent on the model choice, the parameter settings, and so on. A good and intuitive baseline would be a score that could be obtained without the need for tuning parameters or, for that matter, learning anything from the feature values of the input data. Koyejo et al. [49] and Lipton et al. [61] propose optimal threshold classifiers and corresponding baselines for a list of evaluation metrics.

Research Question To determine the threshold classifiers, it is still necessary to learn from the input data. This is not in line with the desire to construct a *general*, *simple*, and *informative* baseline that does not depend on feature values. Hence, we attempt to answer the following RQ:

2*b* *Is there a general, simple, but informative way to benchmark any evaluation metric for binary classification? In other words, can a metric-specific baseline be constructed that any newly developed ML method should clearly outperform?*

This question is the focus of Chapter 5. The answer is provided and explained in Chapter 8.

Contributions We propose a *universal framework* for binary ME, called the *Dutch Draw*, that generates a standardised baseline for many evaluation metrics. It can be seen as the *best dummy classifier* for any given situation. First, the Dutch Draw baseline is general, because it can be used in any binary prediction problem regardless of the domain. Second, it is simple, because it can be calculated very fast, and it is clear that any learning method should outperform the baseline. Third, it is informative, because it is the value of the optimal dummy classifier, and therefore the best score that could be obtained without learning from the features values. Because of these three properties, we advocate for the inclusion of the Dutch Draw baseline in ML research. It gives a strong foundation for assessing the performance of newly developed methods.

1.3 Active Learning

In Section 1.1, we proposed to use a human expert for model evaluation when labelled data is scarce. An example of this is presenting the most anomalous results of an outlier detection technique to an expert and asking whether malicious instances were indeed discovered. The field of ML that is built on this concept is called Active Learning. In this section, we first provide an overview of the general framework of AL. Then, we explain what issues are present within AL and how our approach could aid in mitigating those. Finally, we extend our AL method by incorporating automatic labelling of observations. Hence, we make it possible for ML techniques to actually label the unlabelled data within the AL framework.

1.3.1 Active Learning Framework

The main idea behind AL is that the method can decide which data it wants to be labelled, and consequently, *from which data* it wants to learn [100]. If selection is done effectively, less labelled data is needed to obtain good performance [52]. Figure 1.2 provides an illustration of the AL framework. At the start, a classifier is trained on the current labelled dataset such that it can predict the classes of the observations in the unlabelled pool. Then, the query function makes a selection of one or more instances that are presented to the

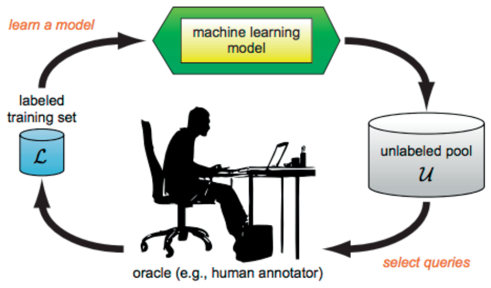


Figure 1.2: Illustration of AL framework (taken from [100])

human expert, the *oracle*. How this query function is defined is usually what characterises the AL approach. There are many query strategies, ranging from simple ones, such as selecting anomalous instances [88], to much more complex ones, such as using Deep Learning [92]. It is generally assumed that the oracle provides the correct labels for the query observations. These instances are then added to the labelled set and removed from the unlabelled pool. Then, the next iteration starts and the procedure repeats itself. Hence, each iteration, the labelled pool becomes larger. If the query function selects the right observations at the right time, performance of the classifier can quickly increase. The procedure terminates when, for example, the *labelling budget* has been exhausted or when some performance score has been reached.

The optimal query function depends on many factors, including the field in which AL is used. In the next subsections, we describe the developments of AL in NID and analyse which query functions have been proposed. We propose an advanced *dynamic query function* that is able to adjust itself during the labelling process. Later on, we investigate how ML can be used as an extra labelling source and propose our simple, yet powerful approach to automatic classification.

1.3.2 Dynamic Active Learning in Network Intrusion Detection

NID is one of the fields in which AL can effectively be applied, because of the difficulty to obtain labelled data [124, 125]. Network connections are diverse, dynamic, and numerous [103]. Hence, several AL approaches have been proposed in cyber security to enhance labelling efficiency. Many of these methods have a query function that focuses on the *prediction uncertainty* of unlabelled data, i.e., the function asks for the labels of observations about which the model is the least certain on how to classify them [36, 39, 60]. Obtaining the correct labels for such data is expected to be informative to the classifier, and hence, prediction performance should increase more than when random observations are queried. Uncertainty selection is one of the possible query strategies, but many more can be constructed. For example, the query function could consist of multiple strategies with each focusing on a different aspect of the data.

Although it seems that for each situation a fitting query function can be constructed, at the start it is not clear which strategy will work well. Considering multiple strategies could improve performance. However, this raises a new question: how to combine the strategies into a better query function? Say we construct a query set of 100 observations, and we want to reserve different parts of the set for specific types of instances. But, which part is assigned to uncertain observations and which part to anomalous instances, for example? Stokes et al. [104] query these types of observations in a 50/50 fashion, but this split is not motivated. Or, maybe we want to combine different data characteristics to obtain a new metric that determines whether an observation is selected. Moreover, when the decision rules have been set at the start, it is not certain that their benefits remain the same throughout the labelling process. Remember that each iteration new query observations are shown to the oracle. Maybe the allocation proportions work well at the beginning, but deteriorate later on in the process.

Research Question We focus on the following RQ:

3*a* *What are the effects on the prediction performance and the labelling process when the query function can dynamically adjust itself to best fit the current situation?*

Chapter 6 supplies the knowledge that is necessary to answer this question. The answer is given and analysed in Chapter 8.

Contributions We introduce *Jasmine*, a novel AL method that allows for changing the allocation balance during the process. As a result, the right type of query observations are queried to the oracle at the right time. We show that Jasmine obtains good and more robust results than several static query functions do. The ability to dynamically update the balance sets our research apart from other AL procedures.

1.3.3 Active Learning with Semi-Supervised Learning

With dynamic updating we optimise the query procedure: the method is able to select the most informative observations to present to the oracle. However, the rate at which the set grows is still limited by the labelling speed of the oracle. Adding another labelling source to the method could speed up the process substantially. Thus, why not use ML for automatic labelling? Besides the small labelled set, we have a rich set of unlabelled data available that can be exploited to classify specific observations without taking resources from the oracle. As before, each iteration a query set is constructed, but now also a subset of unlabelled observations is selected for automatic labelling. A commonly used strategy is based on certainty: if the confidence of the classifier in its prediction exceeds some threshold, then the observation is automatically labelled [115]. Intuitively, by incorporating such a new source of labelling, the labelled pool can expand very quickly, and ideally, performance increases too.

The example provided above is highly reminiscent of SeSL: unlabelled data is used to improve the predictions made by the model trained on labelled observations. Indeed, SeSL has been combined with AL to increase labelling efficiency [40, 56, 115]. However, such Active SeSL (ASeSL) techniques have not been used much in NID. For example, Mao et al. [69] and Zhang et al. [133] introduce methods that assume a distinctive property of the data, but this does not usually hold. Moreover, to ensure whether the data has this property, lots of human effort is necessary [71]. This directly challenges the need to use as little human resources as possible. Lastly, the AL techniques in these ASeSL methods are all of static nature.

Research Question To address the limitations stated above, we pose the following RQ:

3b *What are the effects on the prediction performance when SeSL is used besides AL to automatically label observations?*

This question is tackled in Chapter 7. The answer is provided and discussed in Chapter 8.

Contributions We propose *Plusmine*, a new ASeSL method that incorporates *(i)* our dynamic AL method Jasmine with some improvements, and *(ii)* a novel SeSL approach that allows for automatic labelling. This approach considers the expected effect of automatically classifying data observations on the performance in the next iteration step. This information is then used to determine which observations are actually automatically labelled, and hence, added to the labelled pool of data. Because of its simplicity, it does not require many computational resources, but we show that it does increase the prediction performance more than benchmark methods do.

1.4 Overview of Dissertation

We mold the symbiosis between human and machine by exploring the research topics Anomaly Detection, Model Evaluation, and Active Learning. The contents of Part I of this dissertation correspond to AD, Part II corresponds to ME, and Part III to AL. Table 1.1 gives an overview of the chapters and how they relate to the research questions and the scientific papers and publications.

The specific personal contributions that I made to the scientific papers are listed below.

Table 1.1: Overview of the parts, chapters, papers, and RQs

Part	Chapter	Paper	RQ
I	2	Klein et al. (2018) [47]	1a, 1b, 1c
	3	Mensah, Klein et al. (2019) [72]	1a, 1b, 1c
II	4	Tabatabaei, Klein et al. (2020) [109]	2a
	5	Van de Bijl, Klein, Pries et al. (2022) [118]	2b
III	6	Klein et al. (2022) [45]	3a
	7	Klein et al. (2021) [46]	3b

Ch. 2: Klein et al. (2018) [47]
I took the lead in this project. After the dataset was provided to me, I contributed to the analysis of the data and constructed new features that were expected to aid in prediction. Moreover, I performed the literature review, contributed to the construction of the methodology and experimental setup, and I executed the experiments. I made visualisations of the results and assisted in the interpretations of them. Last, I wrote two versions of the research, the short version was published and the extended version can be read in this dissertation.

Ch. 3: Mensah, Klein et al. (2019) [72]
This research was mainly performed together with Caleb Mensah, an M.Sc. Business Analytics student (now graduated) who I supervised. Together, we conceptualised the project, designed the methodology, constructed the experiments, and analysed the results. I contributed to the literature review and wrote the published paper.

Ch. 4: Tabatabaei, Klein et al. (2020) [109]
This research can be crudely split into two. There is a theoretical and an experimental side. I focused on the first and provided and wrote the mathematical formulation, derivations, and proofs for the research. Furthermore, I offered feedback and adapted other parts of the publication.

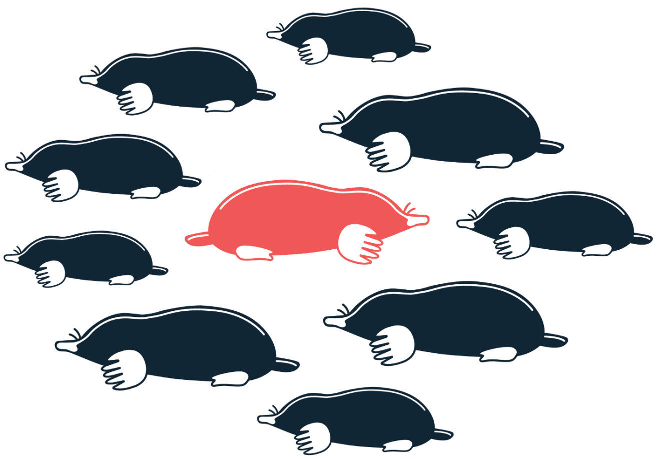
Ch. 5: Van de Bijl, Klein, Pries et al. (2022) [118]
Together with Etienne van de Bijl and Joris Pries, I was actively involved in all research steps. I conceptualised the research topic, contributed to the methodology, mathematical formulations, derivations, and proofs. Moreover, I was involved in composing the experimental setup, and analysing and discussing the results. Lastly, I wrote substantial parts of the scientific paper, provided feedback, and made adaptations to everything.

Ch. 6: Klein et al. (2022) [45]
I had the lead in this research and was involved in all of its components. I reviewed the literature and contributed to the construction of the methodology including the mathematical formulations and derivations. Also, I composed the experimental setup and executed the experiments. After this, I made visualisations of the results and assisted in the analysis of them. The published paper was written by me.

Ch. 7: Klein et al. (2021) [46]
I made contributions to all components of this research. I reviewed the literature and was involved in designing the methodology with all mathematical formulations and derivations. The experiments were composed and executed by me. Moreover, I visualised the results and was involved in the analysis. Finally, I wrote the published paper.

Part I

Anomaly Detection



2 Detecting Network Intrusion Beyond 1999: Applying Machine Learning Techniques to a Partially Labelled Cyber Security Dataset

Ik ben voor de duizendste keer in dit spel heel erg op het verkeerde been gezet.

Paulien Cornelisse
Wie is de Mol? 2013

For decades, Network Intrusion Detection Systems (NIDSs) have been used to defend computer networks against malicious cyber activities. For the construction of such detection systems, different approaches can be taken: aimed at the identification of known attacks, or directed at the discovery of new, previously unknown, intrusions. Both approaches have been widely studied in the field of Machine Learning (ML). However, the second, anomaly-based approach has been employed only recently in practice. This is partly due to the lack of good training data. In most research, a dataset from 1999 is used as an evaluation benchmark, but the problem is that data quickly becomes outdated in the field of cyber security. This research uses a recent, partially labelled dataset based on the Locked Shields 2017 exercise. The main contributions of this chapter are to show how to cope with missing labels, how different ML techniques perform on partially labelled data, and how to determine which features are important. The performance is assessed with the aid of the available labels. Moreover, a cyber security expert analyses the results and shows that the models are able to classify the known intrusions as malicious, and that they can discover new intrusions. In a set of 500 detected anomalies, 50 previously unknown malicious observations are found. Intrusions are not frequent, hence this shows how well an unlabelled dataset can be used to construct and to evaluate an NIDS.

Based on [47]:
Jan Klein, Sandjai Bhulai, Mark Hoogendoorn, Rob van der Mei, and Raymond Hinfelaar
Detecting Network Intrusion Beyond 1999: Applying Machine Learning Techniques to a Partially Labeled Cybersecurity Dataset
2018 IEEE/WIC/ACM International Conference on Web Intelligence

2.1 Introduction

With the continuing rise in the presence of cyber attacks, it becomes more and more important to protect computer networks and data from unauthorised access. On a large scale, the year 2017 has seen some major cyber security disasters [121]. An example is *WannaCry*, a worldwide cyber attack aiming for computers using the Microsoft Windows operating system. It is estimated that more than 200,000 computers in 150 countries were affected with total costs ranging from hundreds of millions to billions of dollars [1].

To discover malicious activities, Network Intrusion Detection Systems (NIDSs) have been developed. There are different types of systems: for example, aimed at *misuse detection* or *anomaly detection* (AD) [103]. The first relies on the properties of known attacks to discover these activities in new network traffic using signature matching algorithms. Misuse detection is effective in recognising previously seen attacks. The second type focuses on activities different from what is expected. A model of normal behaviour is constructed, and hence, deviations from the expected activity are detected. Malicious activities are assumed to exhibit abnormal behaviour, and hence, AD is effective in discovering novel intrusions. Both approaches to network intrusion detection (NID) have been widely researched with the aid of Machine Learning (ML) [131].

However, in the operational setting, AD systems are highly underrepresented. Only recently the use of these systems has moderately increased. Sommer and Paxson [103] argue that this low popularity is due to "*(i)* a very high cost of errors; *(ii)* a lack of training data; and *(iii)* an enormous variability in input data". Misclassification of observations yields a higher cost in intrusion detection than in other fields in which ML is used. Benign observations incorrectly classified as malicious (false positives) burden cyber security specialists with the task to redundantly investigate whether these are dangerous. Yet, especially intrusions labelled as harmless (false negatives) are undesirable, since they can result in serious damage to the network. The lack of good training data is because of the usual sensitivity of the data and the laboriousness of labelling a dataset [68]. There are few publicly available datasets that can be used as a contemporary benchmark to evaluate developed NIDSs. In literature [23, 41, 43, 93, 116], the NSL-KDD dataset is usually used for validation. This is a refined version of a dataset generated in 1999. Since it is almost 20 years old, it is impossible for this dataset to resemble current network traffic. The diversity of network traffic is also a reason for the low presence of anomaly-based NIDSs. Even within a single network, basic features such as the duration of a connection can greatly vary.

The aim of this chapter is firstly to study the performance of different ML techniques on a partially labelled recent dataset. We compare the results of algorithms for Supervised Learning (SL) and Unsupervised Learning (UL). Secondly, this research examines the influence of each feature in detecting cyber attacks. Thirdly, the cyber security expert, who is part of the research team, analyses two samples of observations which have been assigned a high or low probability of being an intrusion by the models and determines whether there are previously unknown malicious activities. Fourthly, the results are compared with a benchmark technique from literature. We use the *Locked Shields 2017* dataset as input data for the different techniques. This dataset contains only positively labelled and unlabelled observations: some activities are known to be malicious, but for the rest of the observations it is unknown whether they are malicious or benign. Before we employ the ML techniques, we examine how we can manipulate the raw Locked Shields data to engineer usable variables for learning. Finally, we demonstrate with our research that it is not necessary to have (completely) labelled data to construct and test an NIDS. Since network intrusion data has to be manually labelled by experts, precious time is saved this way.

The subsequent parts of this chapter are structured as follows. In Section 2.2, we analyse the Locked Shields dataset in its raw form. In Section 2.3, we discuss how relevant features can be engineered and extracted from the data, and we explain how to apply the aforementioned ML techniques. In Section 2.4, we present the results from these different methods and compare them. In Section 2.5, we discuss the implications of the results, draw conclusions, and give suggestions for further research.

2.2 Data: Locked Shields 2017

The unique and recent dataset used in this research is based on the Locked Shields exercise of 2017, organised by the NATO Cooperative Cyber Defence Centre Of Excellence (CCDCOE). Since 2010, the CCDCOE annually organises Locked Shields. This is “the world’s largest and most advanced international technical live-fire cyber defence exercise” [83]. The goal of this exercise is to train the security experts who protect the national IT systems. In short, the teams are given control over a fictional country and are expected to maintain its networks and services. Information about the proceedings during the Locked Shields 2017 exercise is provided by Maennel et al. [67] and experiences of participants of previous years by Kulich [51] and Schuetz and Burschka [98].

The Locked Shields data consists of several log files collected by Bro, which is a network security monitor developed by Paxson [86]. One of the largest files is conn.log, which contains general information on TCP/IP, UDP, and ICMP traffic. This log file consists of $M = 15{,}369{,}736$ observations with 21 variables. On the one hand, not all features are of relevance for the analysis: some of them have a single value across all observations or do not contain relevant information, such as the unique ID of each connection. On the other hand, two extra variables were added by the cyber analyst: subnet_orig and subnet_resp. These variables could play a role in determining whether a connection is an intrusion. This brings the total number of variables to $K = 19$. They are shown in Table 2.1 with a short description of each feature. Lastly, some IP

Table 2.1: Raw features in conn.log

Name	Description	Range and/or size
ts	Unix time of first packet	$\approx [1.49316 \times 10^9, 1.49330 \times 10^9] \subset \mathbb{R}_+$
id.orig_h	source IP address	1,723 addresses
id.resp_h	destination IP address	4,444 addresses
id.orig_p	source port number	$\lvert\{0; \ldots; 65{,}535\}\rvert = 64{,}538$
id.resp_p	destination port number	$\lvert\{0; \ldots; 65{,}501\}\rvert = 5{,}336$
proto	transport protocol	$\lvert\{\mathsf{icmp}, \mathsf{tcp}, \mathsf{udp}\}\rvert = 3$
srv	application protocol	$\lvert\{\text{-}, \mathsf{dhcp}, \ldots, \mathsf{ssl}\}\rvert = 13$
duration	duration connection	$\approx [0; 128{,}232] \subset \mathbb{R}_+$
orig_bytes	# bytes originator	$\approx \{0; 9.806 \times 10^9\} \subset \mathbb{Z}_+$
resp_bytes	# bytes responder	$\approx \{0; 9.284 \times 10^9\} \subset \mathbb{Z}_+$
conn_state	state of connection	$\lvert\{\mathsf{OTH}, \mathsf{REJ}, \ldots\}\rvert = 13$
missed_bytes	# bytes lost in content gaps	$\approx \{0; 1.789 \times 10^8\} \subset \mathbb{Z}_+$
history	history of connection	981 strings
orig_pkts	# packets originator sent	$\approx \{0; 7.452 \times 10^6\} \subset \mathbb{Z}_+$
resp_pkts	# packets responder sent	$\approx \{0; 1.400 \times 10^7\} \subset \mathbb{Z}_+$
orig_ip_bytes	# IP level bytes originator sent	$\approx \{0; 1.002 \times 10^{10}\} \subset \mathbb{Z}_+$
resp_ip_bytes	# IP level bytes responder sent	$\approx \{0; 9.700 \times 10^9\} \subset \mathbb{Z}_+$
subnet_orig	source subnet connection	$\lvert\{\text{-}, \mathsf{BT\_SINET}, \ldots\}\rvert = 26$
subnet_resp	destination subnet connection	$\lvert\{\text{-}, \mathsf{BT01}, \ldots\}\rvert = 23$

addresses appearing in this dataset were indicated as malicious, allowing us to label the corresponding observations as intrusions.

2.3 Methodology

As mentioned before, the Locked Shields dataset is partially labelled: 8,442 ($\approx$ 0.055%) observations are malicious, while the classes of the remaining 15,361,294 ($\approx$ 99.945%) connections are not known. Consequently, both UL and SL techniques were applied, and their results were compared. Semi-supervised algorithms were not considered, because the subset of labelled observations all share the same label (being malicious). The Autoencoder was used as the UL technique to discover the observations that deviate from the rest. The confirmed malicious observations were used as a test for the quality of the model. The Gradient Boosting Machine (GBM) was employed as the SL method. Here, the unknown instances were all considered to be benign. The number of intrusions correctly classified as such gave a measure of performance. Moreover, the important features were determined and the cyber security expert analysed two samples allowing for a different way to assess the accuracy of the models. Finally, the results were compared with those of a benchmark method.

2.3.1 Pre-processing

Before the experiments were carried out, an adequate target dataset had to be assembled. This entailed extracting and engineering the appropriate features using the raw Locked Shields data, and determining how the data could be used for the training, validation, and test phases.

Feature Extraction

An Autoencoder can only handle numerical variables, but as shown in Table 2.1, some variables are categorical. Hence, it was necessary to extract new numerical features using the categorical variables. Since the discussed models were ultimately compared, the constructed numerical dataset was used in both the Autoencoder and the GBM algorithms. One-hot encoding can be effectively used on categorical features: if there are C categories, then $C - 1$ binary variables are introduced with each variable corresponding to a category. The one missing is the *baseline category*. The number of additional variables linearly increases with the number of original categories, hence one-hot encoding was only used on the variables proto and srv with a few categories. Even though conn_state has the same number of categories as srv, it was not split into twelve binary variables. This is because conn_state was used to construct other numerical features that retain its relevant information. This is explained in more detail later.

The variables subnet_orig and subnet_resp indicate from and to which subnet (logical division of an IP network) a connection came and went, respectively. Both features have seventeen categories, which are not necessarily the same. One-hot encoding was also used on these variables.

Feature Engineering

The features id.orig_h and id.resp_h both have many possible categories. This also holds for the port number variables id.orig_p and id.resp_p, since port numbers should be seen as categories even though they are represented by numbers. It would be very ineffective to introduce a binary feature for each of the categories for these variables. Therefore, they were replaced by engineering new features. The previously mentioned NSL-KDD dataset, described in [23], was taken as an example for this process. In that dataset, IP addresses, port numbers, and the connection state are no explicit variables, but numerical features have already been derived from them. For instance, from id.resp_h the new variable resp_h_count_τ was composed. For observation $i \in \{1, \ldots, M\} =: \mathcal{M}$, this variable is the number of connections in the last τ seconds to the same destination host (responding IP address) as that of connection i. Formally,

$$\mathsf{resp\_h\_count\_}\tau_i := |\mathsf{Resp\_h}_i \cap \mathsf{T\_}\tau_i| - 1,$$

where $\mathsf{Resp\_h}_i$ is the set defined as

$$\mathsf{Resp\_h}_i = \{j \in \mathcal{N} : \mathsf{id.resp\_h}_j = \mathsf{id.resp\_h}_i\},$$

$\mathsf{T\_}\tau_i$ is given by

$$\mathsf{T\_}\tau_i = \{j \in \mathcal{N} : \mathsf{ts}_j \in [\mathsf{ts}_i - \tau, \mathsf{ts}_i]\},$$

and the minus 1 is because i itself is always in the connection sets $\mathsf{Resp\_h}_i$ and $\mathsf{T\_}\tau_i$. Equivalently, such features were also composed from id.orig_h, id.orig_p, id.resp_p, proto, and srv.

Next, several new features were engineered from conn_state. This categorical variable has thirteen possible values, which can be partitioned into two groups: (1) a group with normally established connections; and (2) a group with connections that caused some error. According to Dhanabal and Shantharajah [23], the categories S0, S1, S2, S3, and REJ indicate an abnormal state of the connection. However, the official documentation of Bro [114] implies that the categories S0, REJ, RSTR, RSTOS0, RSTRH, SH, and SHR indicate connection states that raised an error. Let Error_state be the set of all observations with one of these error categories. Now, consider for example $\mathsf{resp\_h\_count\_}\tau$, which was just introduced. From this, the variable $\mathsf{resp\_h\_error\_rate\_}\tau$ was constructed, which denotes the fraction of connections inside that count feature which have an error state. More specifically,

$$\mathsf{resp\_h\_error\_rate\_}\tau_i = \frac{|\mathsf{Error\_state} \cap \mathsf{Resp\_h}_i \cap \mathsf{T\_}\tau_i|}{|\mathsf{Resp\_h}_i \cap \mathsf{T\_}\tau_i|}.$$

These features were also constructed for variables id.orig_h, id.orig_p, id.resp_p, proto, and srv. They all take values in $[0, 1]$.

Moreover, we also paired the constructed count variables with each other. For example, let $\mathsf{resp\_h\_same\_proto\_rate}_i$ indicate the fraction of connections in $\mathsf{Resp\_h}_i \cap \mathsf{T\_}\tau_i$ that used the same protocol as i. Hence,

$$\mathsf{resp\_h\_same\_proto\_rate}_i = \frac{|\mathsf{Proto}_i \cap \mathsf{Resp\_h}_i \cap \mathsf{T\_}\tau_i|}{|\mathsf{Resp\_h}_i \cap \mathsf{T\_}\tau_i|},$$

where

$$\mathsf{Proto}_i = \{j \in \mathcal{N} : \mathsf{proto}_j = \mathsf{proto}_i\}.$$

In the NSL-KDD data, some of the aforementioned engineered features were aggregated over a time window of $\tau = 2$ seconds, which gave rise to a short-term analysis of the data. In this research, this was extended to an additional aggregation over $\tau = 120$ seconds (two minutes).

Next, the *Producer-Consumer Ratios* (PCRs) were added to the dataset on the advice of the cyber specialist. These variables describe the interaction between data sent by the originator and by the responder. Generally, the PCR is defined as

$$\mathrm{PCR}_i = \frac{\mathrm{Orig}_i - \mathrm{Resp}_i}{\mathrm{Orig}_i + \mathrm{Resp}_i}.$$

Note that $\mathrm{PCR}_i \in [-1, 1]$, with a positive value meaning that the originator sends more data, a zero value meaning that they send the same amount, and a negative value meaning that the responder sends more. Here, three versions of the PCR were formulated: PCR_bytes, PCR_pkts, and PCR_ip_bytes.

Finally, since relevant information had been derived from id.orig_h, id.resp_h, id.orig_p, and id.resp_p, these variables were removed from the dataset. Also, the features ts and history were discarded, because most of their added values are captured by the variables mentioned before. After the process of feature engineering and extraction, the transformed Locked Shields 2017 dataset consists of $K_{\mathrm{tf}} = 142$ features.

Data Preparation

There were some steps left to do before the experiments could be carried out. Firstly, the first 120 seconds of data were removed. The aggregated temporal variables are skewed at the start of the exercise. This resulted in the removal of 614 instances ($\approx$ 0.004%). Since there are no labelled malicious observations in the first 120 seconds, it was justified to discard them. Secondly, the dataset was randomly split into three independent sets with allocation percentages 70%/15%/15%. The training set (M_{train} = 10,758,386) was used to train the model, the validation set (M_{val} = 2,305,368) to determine the optimal hyperparameters of the ML method considered, and the test set (M_{test} = 2,305,368) to assess the accuracy of the model. Of the total of 8,442 labelled malicious observations, P_{train} = 6,004 ended up in the training set, P_{val} = 1,221 in the validation set, and P_{test} = 1,217 in the test set. Moreover, $K_{\mathrm{train}} = K_{\mathrm{val}} = K_{\mathrm{test}} = 142$.

2.3.2 Autoencoder

The first model considered in this research is a UL method called the Autoencoder. It is a neural network closely related to the multi-layer perceptron with an input layer, at least one hidden layer and an output layer. By definition, the input layer has as many nodes as the output, because the purpose of an Autoencoder is to reconstruct its input. It has been shown that this technique works well for AD and feature dimensionality reduction [32, 132], but not yet in the specific case of NID.

Hyperparameters

During training, the optimal weights are determined such that the loss function is minimised. However, there are hyperparameters which had to be determined beforehand. These were the number of layers $L \geq 3$ in the network, the number of neurons $K_l \in \mathbb{N}$ in layer $l \in \{1, \ldots, L\}$, and the Ridge regularisation shrinkage parameter $\lambda \in \mathbb{R}_+$. The latter parameter was used to avoid overfitting to the data. The activation function used was the hyperbolic tangent function. Note that each additional hidden layer l introduces a new parameter K_l to choose. To reduce this complexity, the condition stating that the network should be symmetric and the geometric pyramid rule were imposed. This rule specifies the number of neurons K_l in layer l by

$$K_l = K_{(L+1)/2} \cdot \left(\frac{K_{\text{train}}}{K_{(L+1)/2}} \right)^{\frac{|2l-(L+1)|}{L-1}}, l \in \{1, \ldots, L\}.$$

Consequently, K_l is a function of the number of neurons in the middle layer $K_{(L+1)/2}$. Hence, the three hyperparameters left were L, $K_{(L+1)/2}$, and λ. Their optimal values were determined with the aid of the validation set. This set was fed to a trained Autoencoder and for every observation the mean squared difference between its input and output was calculated. More specifically, let $x_{i,p}$ be the (standardised) value of feature p for instance i and let $\hat{x}_{i,p}$ be its (standardised) reconstructed value. Then the mean squared error MSE_i is defined as

$$\text{MSE}_i = \frac{1}{K_{\text{train}}} \sum_{p=1}^{K_{\text{train}}} (x_{i,p} - \hat{x}_{i,p})^2. \tag{2.1}$$

The assumption is that a malicious observation cannot be correctly reconstructed, and hence, results in a relatively high MSE_i. The hyperparameter combination $(L, K_{(L+1)/2}, \lambda)$ chosen was the one that maximised the *Discounted Cumulative Gain* (DCG). It is defined as

$$\text{DCG} = \sum_{j \in \mathcal{R}_{\text{val}}} \frac{\text{mal}_j}{\log_2(j+1)}. \tag{2.2}$$

Here, $\mathcal{R}_{\text{val}}$ is the set of ranks of the mean squared errors of the observations in the validation set, hence it is a permutation of $\{1, \ldots, M_{\text{val}}\}$. The instance i with the largest MSE_i obtained rank 1 and the observation with the smallest value got rank M_{val}. For example, the first element of $\mathcal{R}_{\text{val}}$ is the rank of the first element in the set of validation observations. The binary variable mal_j indicates whether the observation corresponding to rank j is malicious (1) or unknown (0). A large value of DCG indicates that the known malicious observations obtained high ranks, and hence, had large MSEs. Note that DCG is bounded: in a 'perfect' ranking all known malicious instances obtain ranks 1 through P_{val} and in the worst ranking all these observations obtain the lowest possible ranks. Formally,

$$\sum_{i=M_{\text{val}}-P_{\text{val}}+1}^{M_{\text{val}}} \frac{1}{\log_2(i+1)} \leq \text{DCG} \leq \sum_{i=1}^{P_{\text{val}}} \frac{1}{\log_2(i+1)},$$

which simplifies to approximately DCG $\in [57.77, 144.94]$ in this case. For convenience, the *normalised DCG* (nDCG) was considered. This is a linearly scaled version of DCG such that it takes values in $[0, 1]$. Now, when the ranks are randomly assigned, then $\mathbb{E}(\text{nDCG}) \approx 0.0532$ and $\text{Var}(\text{nDCG}) \approx 3.68 \times 10^{-6}$. Hence, a value larger than $\mathbb{E}(\text{nDCG})$ means the model performed at least better than random.

Lastly, when no labels are present, it is possible to use a technique such as the elbow method to perform hyperparameter tuning.

2.3.3 Gradient Boosting Machine

The second ML method considered is called Gradient Boosting, or the Gradient Boosting Machine, and can be used for AD [91]. It is a supervised forward-learning technique that constructs a prediction model as an ensemble of decision trees. An observation presented to a trained GBM follows the constructed path through this ensemble and finishes in an end node (or leaf). This node assigns a probability of being in a certain class to the observation. Since Gradient Boosting is an SL technique, it requires labelled training data. Hence, a decision had to be made how to categorise the unknown observations in the dataset. The straightforward approach was to label all unknowns as benign. As a consequence, the malicious class was highly underrepresented ($\approx 0.056\%$).

Hyperparameters

Some parameters had to be determined before performing Gradient Boosting: the number of trees $T \in \mathbb{N}$ to be constructed and the class sampling factors $\alpha_0, \alpha_1 \in \mathbb{R}_+$. Class $c \in \{0, 1\}$ is undersampled if $\alpha_c < 1$ and oversampled if $\alpha_c > 1$. The optimal values for these hyperparameters were determined by the validation set similar to what was done for an Autoencoder. Each observation i in the validation set was fed to the trained GBM (given some parameter combination) and the probability $p_i^{(1)}$ of being in class 1 was obtained. The probabilities of all validation instances were ranked and the DCG as defined in Equation (2.2) was calculated (with $\mathcal{R}_{\text{val}}$ the set of ranks of the probabilities).

2.4 Results

The procedures described in Section 2.3 were conducted in R with the aid of the cluster computers at SURFsara in Amsterdam. Both ML techniques are available in the H2O package, which was developed by H2O.ai. This is

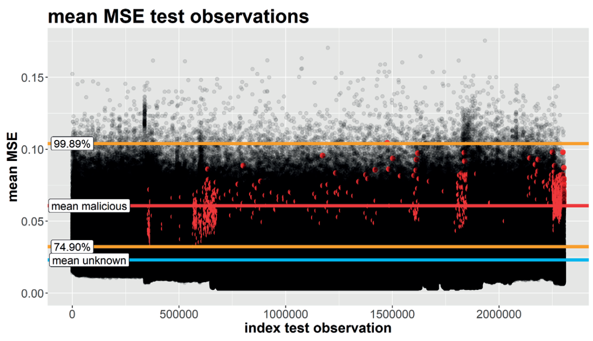


Figure 2.1: Mean MSE of test observations

an open source platform enabling the use of several ML algorithms. In the benchmark evaluation the C5.0 algorithm was used, which is available in the `C50` package.

2.4.1 Results Autoencoder

The training set was used to construct an Autoencoder network. The purpose of an Autoencoder is to design a representation of normal behaviour, thus the 6,004 assumed to be abnormal known intrusions (0.056%) were removed from the dataset. Consequently, the algorithm did not train on the labelled observations. However, there are still possibly many unknown intrusions in the training set.

Hyperparameter Tuning

Several Cartesian grid searches over the hyperparameters were used to maximise the discounted cumulative gain $\mathrm{DCG}_{\mathrm{auto}}$ (as defined in Equation (2.2)). The initial values of the weights in an Autoencoder were $\mathcal{N}(0,1)$-distributed. In other words, training an Autoencoder is a stochastic process, and so, the performance measure is a random variable. Therefore, 60 Autoencoders were trained for each hyperparameter combination, which gave an acceptable estimate for the expected training procedure. The search over the number of layers L ranged over $\{3,5,7,9,11\}$; the search over the number of middle neurons $K_{(L+1)/2}$ over $\{2,5,10,15,20,25,30,40,50,60,70\}$; and the search over the shrinkage parameter λ over $\{\{10^{-j}\}_{j\in\{2,\ldots,13\}},0\}$. Combinations with a small L and/or $K_{(L+1)/2}$ are preferable, because these convey simpler neural networks. The combination $(L, K_{(L+1)/2}, \lambda) = (9, 15, 10^{-4})$ yielded the relatively best performance on the validation set with sample mean $\overline{\mathrm{nDCG}_{\mathrm{auto}}} \approx 0.230$ and sample variance $s^2(\mathrm{DCG}_{\mathrm{auto}}) \approx 0.00139$.

Table 2.2: Five most important features by Autoencoder

feature	share
subnet_orig	8.14%
srv	6.74%
resp_p_same_srv_rate_120s	4.88%
resp_p_same_srv_rate_2s	3.70%
subnet_resp	3.69%

Evaluation on Known Intrusions

The test set was used for the evaluation. An Autoencoder was trained (with the determined hyperparameters by the validation set) on the training set, and applied to the test set, resulting in a reconstruction MSE for each test observation. This procedure was repeated 50 times to obtain 50 MSEs for each test instance. The mean was taken over all these repetitions to estimate the expected reconstruction MSE. The result is shown in Figure 2.1, where the mean MSE[1] of every test observation is plotted. The black points are the unknown instances, while the red points are the labelled intrusions. Solid black regions correspond to a high density of unknown instances. These are mostly found in the lower parts of the plot, indicating that most of the observations have a relatively small MSE. The lowest orange line indicates the *intrusion infimum*: the level such that all known intrusions are above this line. Hence, the lowest 74.90% of the data do not contain labelled intrusions. Likewise, the highest orange line is the *intrusion supremum*: the level such that all labelled malicious instances are below this line. This means that there are no known intrusions in the highest 0.11% of the data. Moreover, the blue line indicates the mean MSE of the unknown instances (with value 0.0230) and the red line that of the labelled observations (with value 0.0607). Finally, the average Autoencoder had $\text{nDCG}_{\text{auto}} \approx 0.176$.

Feature Analysis

The MSE for each test observation was calculated by averaging over the per feature squared errors (see Equation (2.1)). To determine which variables are important for the Autoencoder, the share of each feature in the total squared error (TSE) was calculated, where TSE is the sum of all squared errors. The larger the share of a feature, the higher the importance of that variable. The five most important features for the average labelled intrusion are shown in Table 2.2. Here, the features which were partitioned into binary variables during one-hot encoding were unified again.

Table 2.3: Analysis by the expert

	benign	**malicious**	**unknown**
high	65.4% (327)	10.8% (54)	23.8% (119)
low	98.6% (986)	0.1% (1)	1.3% (13)
class total	87.53% (1,313)	3.67% (55)	8.80% (132)

Evaluation by Expert

The cyber security expert analysed two samples of test observations. The 'high sample' contains 500 randomly selected observations from the top 6% of test records which yielded the largest MSEs, so from the 138,322 highest black points in Figure 2.1. The (relatively) 'low sample' is a randomly selected set of 1,000 observations from all test records excluding the top 6%. The analysis by the specialist is summarised in Table 2.3. He was able to classify most of the observations in the two samples: 132 of them were difficult to identify and require deeper inspection. The accuracy of the Autoencoder could be assessed in a new way using this expert classification. Precision, recall, and their harmonic mean the F_1 score were used for this purpose. Since the Autoencoder algorithm is unsupervised, some threshold had to be imposed such that all sample observations with an MSE larger than this value were predicted to be malicious. The best results were obtained when $\alpha = 5.76\%$ of the data was assumed to be anomalous. All three of the measures were maximal for this α with precision 0.142, recall 0.982, and F_1 score 0.248.

2.4.2 Results Gradient Boosting Machine

For Gradient Boosting the complete training set with benign and malicious observations was used. As mentioned before, the class of intrusions is under-represented in the dataset, which is an obstacle for a GBM. It is an SL technique, and so, it uses the given labels to learn to predict the classes of newly presented observations. Therefore, the classes had to be balanced to some extent. The unknown observations could be undersampled or the malicious instances could be oversampled to enable this. The first solution was selected because of three reasons: *(i)* the zero-class actually consists of unknown traffic, hence if there are some unidentified malicious observations mixed with benign traffic, then by subsequently sampling a small portion, these are often skipped and do not play a large role in the construction of every decision tree; *(ii)* oversampling the intrusion class (one-class) implies that the algorithm severely trains on the properties of the specific known intrusions, while the zero-class is much more diverse; and *(iii)* oversampling the one-class to balance both classes means blowing the 6,004 intrusions up to 10,752,382 observations, which has an undesirable effect on the computation time.

[1]In the following, 'mean' in 'mean MSE' is omitted for readability. Hence, whenever 'MSE' is written, this actually implies 'mean MSE'.

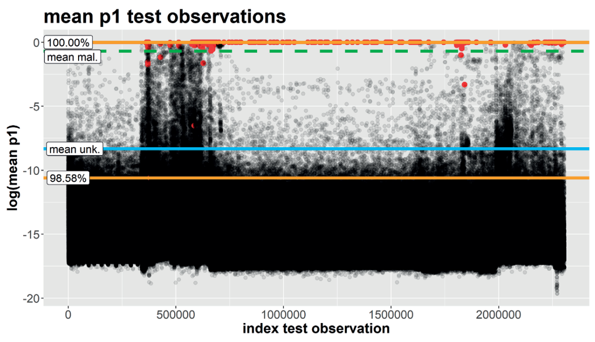


Figure 2.2: log mean intrusion probability of test observations

Hyperparameter Tuning

To find the hyperparameter combination that maximised the discounted cumulative gain $\mathrm{DCG}_{\mathrm{GBM}}$ (defined in Equation (2.2)), several Cartesian grid searches were performed over the number of trees

$$T \in \{10, 20, 40, 80, 150, 300, 450, 600, 750, 900\},$$

and the zero-class sampling factor

$$\alpha_0 \in c_{\mathrm{train}} \cdot \left\{ \frac{1}{3}, \frac{2}{3}, 1, \frac{4}{3}, \frac{5}{3}, 2, \frac{5}{2}, 3, \{4 \cdot 2^k\}_{k\in\{0,\ldots,8\}} \right\},$$

with $c_{\mathrm{train}} = P_{\mathrm{train}}/(M_{\mathrm{train}} - P_{\mathrm{train}})$ the ratio between the number of malicious and unknown observations in the training set. Similar to the Autoencoder, Gradient Boosting is a stochastic process: even when the parameters are fixed, the results differ per training round. To estimate the expected value of $\mathrm{nDCG}_{\mathrm{GBM}}$, 60 GBMs were trained per hyperparameter combination. The combination $(T; \alpha_0) = (300; c_{\mathrm{train}} \cdot 8)$ yielded the relatively best performance on the validation set with sample mean $\overline{\mathrm{nDCG}}_{\mathrm{GBM}} \approx 0.992$ and sample variance $s^2(\mathrm{nDCG}_{\mathrm{GBM}}) \approx 9.38 \times 10^{-6}$.

Evaluation on Known Intrusions

As before, the test set was used for the evaluation. Gradient Boosting was applied to the training set with the determined hyperparameters. The acquired model was used on the test set, resulting in an intrusion probability for each observation. Since the training process is stochastic, it was repeated 50 times. However, it is not correct to simply take the mean of the realised intrusion probabilities for a test observation as an estimate for its expected probability. In Gradient Boosting, the predicted label is not necessarily 1 when the corresponding probability is at least 0.5 and 0 otherwise. There is some threshold value $\theta \in (0, 1)$ (based on the F_1 score) that determines which label is predicted. This value is different per training procedure. To allow for comparison

Table 2.4: Five most important features by GBM

feature	share
resp_p_same_srv_rate_120s	52.89%
resp_p_same_resp_h_rate_120s	16.19%
srv	8.00%
subnet_orig	7.69%
srv_same_resp_p_rate_120s	1.89%

of the different runs, each run the threshold θ was transformed to be 0.5 and the probabilities were changed accordingly. This was done by applying the function $f_\theta : [0,1] \to [0,1]$ given by

$$f_\theta(p) = \frac{(1-\theta)p}{(1-2\theta)p+\theta}$$

to all probabilities. This function has desirable properties: *(i)* it is continuously differentiable, *(ii)* $f_\theta(0) = 0$, *(iii)* $f_\theta(\theta) = 0.5$, *(iv)* $f_\theta(1) = 1$, and *(v)* $f'_\theta(p) \geq 0$. The resulting mean intrusion probabilities[2] are shown in Figure 2.2. Note that the natural logarithm of the probabilities was taken to enhance the amount of information the plot conveys. The black points are the unknown observations and the red points are the labelled malicious instances. As before, the orange lines correspond to the intrusion infimum and supremum. The infimum is the 98.58th percentile of the test data and the supremum is the (almost) 100th percentile. The mean of the unlabelled records is 2.44×10^{-4} (log value: -8.32) and is represented by the blue line. The red line representing the mean of the known intrusions, with value 0.98 (log: -0.017), is hidden behind the orange supremum line. The green dashed line corresponds to probability 0.5 (close to the 99.93th percentile) and indicates the border between the predicted classes. Finally, the average GBM had $\text{nDCG}_{\text{GBM}} \approx 0.993$.

Feature Analysis

The importances of the variables for the GBM were determined by the function h2o.varimp. As mentioned before, the features to which one-hot encoding was applied were unified. The five most important variables are shown in Table 2.4.

Evaluation by Expert

Additionally, the accuracy of the GBM was determined by the analysis of the two samples by the cyber security expert. Gradient Boosting is an SL technique, so the label of each test observation was explicitly predicted. The GBM classified $\alpha = 0.0668\%$ of the records as malicious. Comparing the predictions

[2]In the following, ‘mean’ in ‘mean probability’ is omitted for readability. Hence, whenever ‘probability’ is written, this actually implies ‘mean probability’.

to the actual labels assigned by the cyber analyst resulted in a recall of 0.0727, precision of 1.00, and F_1 score of 0.136. Note that the composition of the samples was determined by the results of the Autoencoder. Due to constraints on the availability of the specialist, it was not possible to select two new samples based on the GBM.

2.4.3 Results Benchmark

To assess how well the methods discussed in this research performed with this feature set, the obtained results were compared with a benchmark method. Some features engineered using the raw data were based on the work of Dhanabal and Shantharajah [23]. One of the classification techniques that they used on the NSL-KDD dataset was the C4.5 decision tree algorithm. Here, the improved C5.0 algorithm [95] was applied to the Locked Shields dataset with a selected subset of variables to match the set used by Dhanabal and Shantharajah [23] as good as possible. Their feature set does not contain the variables suggested by the cyber security expert, the variables aggregated over 120 seconds, and the variables paired with the constructed count variables (see Section 2.3.1 for details). This reduced the number of features from $K_{\text{tf}} = 142$ to $K_{\text{bench}} = 39$.

Evaluation on Known Intrusions

Since C5.0 is an SL technique, all unknown observations were considered to be benign. Moreover, the hyperparameters were based on those selected for the GBM. This essentially means that the number of benign instances to be sampled was chosen to be eight times the number of malicious observations P_{train}. Again, because of randomness, 50 models were trained and the mean intrusion probability was calculated for each test observation. This resulted in the average performance measure $\text{nDCG}_{\text{bench}} \approx 0.956$.

Evaluation by Expert

The performance of the C5.0 method was also evaluated by the cyber analyst. This technique classified $\alpha = 0.403\%$ of the test observations as malicious with a precision of 0.571, recall of 0.0727, and F_1 score of 0.129.

2.5 Discussion

2.5.1 Interpretation of Results

The results presented in Section 2.4 showed that the two discussed ML methods performed quite differently. At first glance, the GBM seems better. Figure 2.2 shows that there is a clear distinction between the known malicious observations and the unlabelled connections. Most of the test intrusions are above the green dashed line, and hence, were classified as threats by the GBM. All known

Table 2.5: Accuracy measures on test samples

	nDCG	α	**precision**	**recall**	F_1 **score**
Autoencoder	0.176	5.76%	0.142	0.982	0.248
GBM	0.993	0.0668%	1.00	0.0727	0.136
benchmark	0.956	0.403%	0.571	0.0727	0.129

attacks were in the top 1.42% of the data, as the orange infimum line shows. Also, the means of the probabilities of the two classes were vastly different: for the unknown class it was close to 0, while for the intrusion class it was close to 1. In general, the Autoencoder was somewhat able to notice that the labelled malicious activities do not conform to the normal behaviour of the network traffic, because all test intrusions yielded MSEs in the top 25.10%, as can be seen in Figure 2.1. Unfortunately, this result is not desirable when dealing with millions of observations. This was also reflected in the rather small discounted cumulative gain: $\text{nDCG}_{\text{auto}} \approx 0.176$. There was a distinction between the means of both classes, however. The mean of the unknown class was the 62.00th percentile, while the mean of the intrusion class was the 95.02th percentile.

This is not the whole picture, though. As mentioned before, only some intrusions were explicitly labelled as such, but the rest of the data was unlabelled. The previous paragraph focused on the performance of the two methods with the assumption that all unlabelled observations were actually benign. To obtain a better insight, two samples were taken from the results of the Autoencoder and analysed by the cyber security expert. As Table 2.3 shows, one new malicious activity was discovered in the low sample and no fewer than 54 intrusions were found in the high sample (four of them were already known). Table 2.5 gives a summary of the results obtained in this research. It shows that almost all the real attacks in the two samples were found by the Autoencoder (recall = 0.9818). However, only a small fraction of the anomalies was actually malicious (precision = 0.1417). For the GBM, it is the other way around: almost none of the actual intrusions were found (recall = 0.07273), only the four previously known attacks were correctly classified; and all predicted intrusions were in fact malicious (precision = 1). Consequently, the GBM generated a lot fewer false positives ($\alpha_{\text{GBM}} \ll \alpha_{\text{auto}}$), but it was not able to discover new threats in the data. Additionally, the GBM outperformed the benchmark method C5.0 with respect to generating fewer false positives with the same number of false negatives. Therefore, it had the same recall, but a larger precision.

The addition of the features as discussed in Section 2.3.1 is justified by the results of the benchmark algorithm. The C5.0 algorithm yielded $\text{nDCG}_{\text{bench}} \approx 0.956$ on a subset of the features, but the GBM resulted in $\text{nDCG}_{\text{GBM}} \approx 0.993$ on the complete feature set. Although the values seem close, the difference between the two supervised algorithms is statistically significant: performing a one-sided Mann-Whitney U test on the two samples of DCGs resulted in a p-value smaller than 2.2×10^{-16}, strongly indicating that DCG_{GBM} is larger than $\text{DCG}_{\text{bench}}$. Moreover, the feature analyses of both the Autoencoder and the GBM showed that the features introduced in this research are important in detecting and discovering intrusions, as can be seen in Tables 2.2 and 2.4. The variables aggregated over a time horizon of 120 seconds, the variables added by the expert, and the extra variables which were not present in the research by Dhanabal and Shantharajah [23] all had a relatively large influence on the classification of the test observations.

2.5.2 Conclusion

The aim of this chapter was to study the performance of different ML methods on a partially labelled recent dataset, and to determine the relevance of the features. Firstly, we discussed how reasonable features could be extracted and engineered using a cyber dataset, and we showed that these features are indeed important in the detection of known and novel intrusions. Secondly, our research suggested that NIDSs do not need completely labelled datasets, which hopefully encourages the use of recent datasets to properly assess the quality of detection systems. The GBM showed that it is able to correctly classify the already known intrusions. The Autoencoder, on the other hand, showed that it can be used in detecting new malicious observations. However, it needs to be stressed that a lot of false positives were generated. A possible way to reduce their prevalence is to consider the dependency between the records, since the network connections in the dataset are interconnected. This means that when an observation is deemed to be an anomaly by the NIDS, but the rest of the associated connections are not, then it probably is a false positive. This does not explicitly reduce the number of false positives, but it does save cyber analysts time.

One of our suggestions for future research is the use of Active Learning (AL) [36]. With such methods the experiments performed here can be streamlined, since we 'manually' used the information of the labelled intrusions to improve the performance of the models. AL incorporates this information into its models. We focus on this in Part III of this dissertation.

3 Detecting Fraudulent Bookings of Online Travel Agencies with Unsupervised Machine Learning

Effe tegen je ethische kompas in, kom!

Fresia Cousiño Arias
Wie is de Mol? 2022

Online fraud poses a relatively new threat to the revenues of companies. A way to detect and prevent fraudulent behaviour is with the use of specific Machine Learning (ML) techniques. These anomaly detection (AD) techniques have been thoroughly studied, but they are not employed as much. The airline industry suffers from fraud by parties such as online travel agencies (OTAs). These agencies are commissioned by an airline carrier to sell its travel tickets. Through policy violations, they can illegitimately claim some of the airline's revenue by offering cheaper fares to customers.

This research applies several AD techniques to detect fraudulent behaviour by OTAs, and assesses their strengths and weaknesses. Since the data is not labelled, it is not known whether fraud has actually occurred. Therefore, unsupervised ML is used. The contributions of this chapter are, firstly, to show how to shape the online booking data, and how to engineer new and relevant features. Secondly, this research includes a case study in which domain experts evaluate the detection performance of the considered ML methods by classifying a set of 75 bookings. According to the experts' analysis, the techniques are able to discover previously unknown fraudulent bookings, which will not have been found otherwise. This demonstrates that AD is a valuable tool for the airline industry to discover fraudulent behaviour.

Based on [72]:
Caleb Mensah, Jan Klein, Sandjai Bhulai, Mark Hoogendoorn, and Rob van der Mei
Detecting Fraudulent Bookings of Online Travel Agencies with Unsupervised Machine Learning
2019 International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems

3.1 Introduction

Since industries have expanded their services to the internet to reach more customers, new ways have evolved to claim part of a company's revenue. Aviation faces a considerable problem with these malpractices. In 2008, airline industries all over the world missed out on 1.4 billion US dollars due to fraud. This was around 1.3% of their total revenue, although the rates were up to 4% in parts such as the Middle East and Latin America. Nowadays, these figures are expected to be even higher [12]. One of the conductors of fraud in the airline industry are online travel agencies (OTAs). Such an agency specialises in selling travel products including flights, hotels, and rental cars to customers online. There is a wide variety in OTAs, but they all share at least one similarity: they have an agency agreement with the supplier to resell its products [94]. In this case, the airline carrier allows the OTA access to its booking system to sell aeroplane seats. This expands the reach of the carrier, and therefore, increases its revenue. However, some OTAs violate the policies conducted by the airline organisation in order to get access to cheaper ticket fares. This is possible, because the price of an aeroplane seat depends on several well-known factors. These include the seat's class (economy or business), the flight destination, and the remaining time until departure. More specifically, when a flight consists of multiple flight segments, the price of a single segment can differ depending on the other segments in the complete flight. Here, a flight segment can be seen as the part between the departure and arrival of an aeroplane. If it lands more than once, there are multiple flight segments. An OTA can add one or more artificial segments to a flight to possibly get access to relatively lower prices. Later on, it can cancel these segments, which leads to revenue loss for the airline company.

In general, fraudulent behaviour is assumed to be unusual, and hence, (largely) deviates from the expected, normal behaviour. A way to discover such anomalous behaviour is with the use of outlier detection techniques. Usually, the data with potentially fraudulent behaviour is unlabelled, suggesting the use of Unsupervised Learning (UL). This can be applied in a wide variety of domains, such as insurance, health care, and cyber security (see Chapter 2), with the same goal of finding malicious activities in data [14]. However, most of the applications are to discover and prevent bank fraud. For example, Bolton and Hand [8] propose the use of unsupervised profiling methods to detect credit card fraud in financial transactions on a customer-based, while Ferdousi and Maeda [27] examine the occurrence of fraud in stock market data as anomalous behaviour in an evolving time series.

In the airline industry, the data consists of flight bookings, which can be seen as customer-based data changing through time. However, there are some important differences between bookings and financial transactions. First of all, customers are usually not aware of an OTA conducting fraud and are not directly affected by it. Fraud can even be advantageous to the customer who can purchase a cheaper flight ticket. Furthermore, a booking can be fraudulent because of how it changes through time, in contrast with fraud in a single financial transaction. Lastly, OTAs are part of the business model and are necessary for the airline carrier to make a profit. Of course, the majority of them act sincerely.

Since the airline industry has some characteristics that set it apart from other fields in which fraud occurs, it is interesting to examine how anomaly detection (AD) methods perform. More importantly, we were not able to find literature on the detection of fraudulent behaviour of OTAs. This chapter addresses that research gap. The contributions of our research are, firstly, to show how three different algorithms are applied to the booking data of OTAs to discover violations of the policies conducted by the airline carrier. This allows us not only to eventually block fraudulent bookings, but this can also enrich domain experts with new knowledge on how to avoid malicious behaviour from happening. Before the techniques are applied, practically usable data is constructed from raw booking datasets. To this end, existing features are modified and new variables are added. Secondly, we show the importance of the engineered features in discovering fraudulent bookings. An evaluation set of 90 bookings is constructed for domain experts to classify as either normal or fraudulent. We assess how well the AD methods are able to find these fraudulent observations.

This rest of this chapter has the following structure. In Section 3.2, we examine the characteristics of the raw data and how it can possibly be used to detect fraudulent bookings. In Section 3.3, we transform the data into one usable dataset by means of feature engineering and either normalisation or standardisation, and we introduce the AD methods. The setup of the experiments is given in Section 3.4 and their results in Section 3.5. In the latter, several evaluation metrics are provided for the set of samples that were labelled by the domain experts. Also, features that turned out to be important in detecting fraud are discussed. Finally, in Section 3.6, we draw conclusions about the research and discuss directions for future work.

3.2 Data

The data used in this research was obtained from an airline company. It has several kinds of features. The first type is based on the passengers' travel requirements information, summarised as a passenger booking. It consists of features such as travel dates, travel routes, ticket information, and associated OTAs for all flights planned for the coming 360 days. There were some observations with missing values for some features. We decided not to remove all of them, since having missing values in certain fields could be related to fraudulent behaviour of OTAs. Missing information could be due to an error in the reservation system, which could have been exploited by an OTA. The second type of features contains information about revenue for each created booking. Actual revenue data is only available for ticketed (paid) reservations, while it is estimated for non-ticketed reservations using historical revenue data. The third type of variables is directed at the OTAs themselves. It provides characteristics such as a unique identifier and their location (or market).

The goal is to find fraudulent bookings and the corresponding OTAs that violate the policies of the airline carrier. The observations in this raw dataset were given on a flight segment level. However, the data needs to be booking-based, i.e., each observation should indicate a booking. Therefore, the flight segments corresponding to the same booking had to be merged.

3.3 Methodology

In this section, we discuss how the segment-based raw data was merged and how new variables were constructed from the raw features. Furthermore, we provide a preliminary analysis of the data and introduce which ML techniques were used for the experiments. Lastly, we discuss some transformations that were applied to the dataset with the goal to improve the results.

3.3.1 Feature Engineering

Before the experiments were carried out, new variables that better represent the underlying characteristics of the data were engineered using the raw features that were described in Section 3.2. They can be categorised into two classes: *(i)* revenue-based features, and *(ii)* booking-based features.

Revenue-based Features

The first category of features was derived from the variables containing revenue information. These new features were introduced to describe the relative amount of revenue generated per booking and to compare the expected revenue with the ticketed revenue received. The predictions in revenue were based on a historical horizon of fifteen days that was advised by domain experts. They expected the majority of the changes to occur during this time window. Moreover, the *predicted minimum* and *predicted maximum* revenue were added as features, and a feature describing the *changes in revenue* over the time horizon. A relatively large difference between the predicted maximum and actual revenue could indicate malicious booking behaviour. Since these new features were obtained per flight segment, the records corresponding to the same booking were aggregated (by taking both the sum and average) to obtain one observation for each feature per booking. Furthermore, the *ticketing time* and a feature describing the *variation in ticketing times* for the flight segment were included. The ticketing time is the time it takes before a booking has been paid for. When a flight is legitimately booked online, the payment is expected to be done directly for the whole flight, and hence, there should be no or only a small variation in the ticketing times of the flight segments. A relatively large variation could indicate fraudulent behaviour.

Booking-based Features

The second class of features was derived from the raw booking features. These new features do not only describe the important characteristics of the booking, but they also represent the OTA providing the flight ticket. As mentioned before, flight segments corresponding to the same booking were aggregated to obtain one observation per booking. A new feature of interest is *point of commencement (PoC) circumvention*. This feature checks whether the effective PoC is equal to the true PoC. Here, the effective PoC is the starting point the passenger is expected to depart from, while the true PoC is the actual starting point. PoC circumvention occurs when a fake flight segment is added to a booking to get access to a cheaper fare. Before the aeroplane departs, this flight segment is cancelled while the lower flight price is retained. A difference between the effective and true PoC of a booking coincides with one or more cancellations or additions of flight segments, so features were added which explicitly indicate this behaviour. This was done by comparing booking data on successive days and by calculating the differences in the number of flight segments in each booking. Furthermore, variables which indicate whether an OTA chronologically books flight segments (from first departure until last arrival) were included. These features are directly linked to policy violations. Lastly, several other features were composed that capture other booking related data, such as the *number of passengers* in a booking, the *length of stay*, *number of days between cancellations*, and so on.

Final Dataset

After the feature engineering process, the final dataset consists of $K = 84$ numerical features on $M = 17{,}886$ unique bookings. The total number of unique OTAs is 158. Now, anomalies can be found on a booking level. Each booking is connected to an OTA, making it possible to find the agencies that were potentially conducting fraud.

3.3.2 Feature Analysis

Before the ML algorithms were applied, a preliminary feature analysis on fifteen days of data was performed. The purpose of this study is to give an insight into the booking data in the considered airline market and to examine what fraudulent behaviour of an OTA could be. The features of interest in this exploratory study were those that are concerned with PoC circumvention.

To this end, the cancellations made in the bookings were examined. Table 3.1 shows the adjustments made in several bookings. These modifications were not just caused by cancellations, but also by the addition of new flight segments. This occurs in, for example, the last row. It shows an increase in the number of flight segments on day 12 and a decrease (cancellation) two days later on day 10, which is odd. This process repeats itself on day 5. It is unlikely that a passenger made such adjustments. A deeper analysis of a booking with

Table 3.1: Number of flight segments in six different bookings for the past fifteen days

Days from now into the past														
14	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
7	7	7	7	7	7	7	7	7	6	6	6	6	6	6
6	6	7	7	7	7	7	7	7	7	7	6	6	6	6
5	5	5	5	5	5	5	5	5	5	5	5	5	5	6
7	7	7	7	7	7	7	7	7	7	7	7	7	6	6
2	2	2	2	2	2	2	2	2	2	4	4	2	2	2
6	6	7	7	6	6	6	6	6	7	7	6	6	6	6

Table 3.2: Description of a particular booking on the first and second day. The columns indicate the departure date, segment identifier, departure location, arrival location, effective PoC, true PoC, and PoC circumvention, respectively

Booking properties on day 1						
Day	**ID**	**Dep.**	**Arr.**	**Eff. PoC**	**True PoC**	**PoC circumv.**
0	1	L_1	L_2	NA	PoC_1	NA
0	2	L_2	L_3	NA	PoC_1	NA
5	3	L_3	L_4	NA	PoC_1	NA
5	4	L_4	L_1	PoC_1	PoC_1	0
Booking properties on day 2						
Day	**ID**	**Dep.**	**Arr.**	**Eff. PoC**	**True PoC**	**PoC circumv.**
5	3	L_3	L_4	NA	PoC_2	NA
5	4	L_4	L_1	PoC_1	PoC_2	1

cancelled flight segments is shown in Table 3.2. Here, the two tables represent the same booking, but on different days. Note that the first two rows on the first day are not present on the second day any more: these flight segments have been cancelled. It is interesting to note that the values of the second column, the segment identifier, were not adjusted when flight segments were deleted. After examining this for several bookings with cancellations, it was concluded that the segment identifier was never modified. Hence, unexpected behaviour in that variable could indicate this kind of fraud.

An overview of descriptive PoC characteristics is given in Table 3.3. Here, the

Table 3.3: Overview of the descriptive statistics in the PoC features

Description	**Value**
Percentage of PoC circumvented flight segments	7.27%
Number of unique effective PoCs	115
Number of unique true PoCs	138

percentage of flight segments with PoC circumvention is around 7%. Moreover, the table shows that the number of unique true POCs is greater than the number of effective PoCs. This difference indicates that there are at least 23 locations being used to circumvent the availability.

3.3.3 Anomaly Detection Techniques

Three AD techniques were considered in this research: Isolation Forest (IF), one-class Support Vector Machine (ocSVM), and k-means clustering. These methods were chosen such that we consider a wide variety of AD techniques. Since no labelled data is available, UL methods were used. They assume that the majority of the observations is normal, while only a small fraction is abnormal. This is the case for fraudulent bookings in the airline industry.

Isolation Forest

The first UL technique was designed by Liu et al. [65]. In contrast to traditional AD methods, an IF explicitly separates anomalies rather than determining normal behaviour and identifying anomalies as deviations from that behaviour. This algorithm is more effective and efficient in detecting anomalies than commonly used distance- and density-based methods [65]. In short, an IF determines how long it takes for each observation to be separated, which is done by continuously splitting features between their minimum and maximum values. Since the splits are performed on a feature level, the importance of each feature can be easily derived. Each tree $t \in \{1, \ldots, T\}$ in an IF of $T \in \mathbb{N}$ trees yields a path length $h_t(\mathbf{x}_i)$ for every observation $\mathbf{x}_i \in \mathbb{R}^K$, $i \in \{1, \ldots, M\}$, with K the number of features and M the total number of observations. Anomalies are the records with the smallest average path lengths, because they can be isolated rapidly.

There are two hyperparameters in an IF: the sub-sampling size $\psi \in \mathbb{N}$, and T. The first parameter controls the training data size per tree, while the second one determines how many isolation trees are constructed during training. The *anomaly score* $s_N(\mathbf{x}_i)$ determines how anomalous observation $\mathbf{x}_i$ is. It is defined as

$$s_M(\mathbf{x}_i) = 2^{-\frac{\overline{h}(\mathbf{x}_i)}{c(M)}} \in (0, 1),$$

where $\overline{h}(\mathbf{x}_i) = (1/T)\sum_{t=1}^{T} h_t(\mathbf{x}_i)$ is the average path length of $\mathbf{x}_i$ in the IF, and $c(M) = 2H_{M-1} - 2(M-1)/M$ is the expected path length with H_n the n-th harmonic number. Liu et al. [65] offer some rules of thumb: if $s_M(\mathbf{x}_i) \gg 0.5$, then $\mathbf{x}_i$ can be considered as an anomaly; if $s_M(i) \ll 0.5$, then $\mathbf{x}_i$ can be regarded as normal; and if $s_M(\mathbf{x}_i) \approx 0.5$, then the status of $\mathbf{x}_i$ is vague.

One-class Support Vector Machine

The second UL method applied in this research was designed by Schölkopf et al. [97]. The goal of this Support Vector Machine (SVM) is to identify one specific class amongst all observations. This results in trying to separate the observations belonging to the normal class from the rest of the feature space. Hence, the instances that do not lie within the non-linear normality boundary are considered to be anomalous. Therefore, ocSVM is a boundary-based algorithm. Tax and Duin [113] showed that such algorithms perform better than density-based techniques, since they solve a fundamentally easier problem. Consequently, ocSVM is widely used in the field of AD.

The goal of ocSVM is to separate the data from the origin with maximum margin. A quadratic program is solved to determine the normality boundary, yielding an optimal normal vector $\mathbf{w}$ and margin ρ. There is a hyperparameter $\nu \in [0, 1]$ acting as a trade-off between the fraction of anomalies in the data and the number of training examples used as support vectors [42]. The anomaly score $s(\mathbf{x}_i)$ of an observation $\mathbf{x}_i$ is given by

$$s(\mathbf{x}_i) = \operatorname{sgn}((\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}_i)) - \rho),$$

where $\mathbf{\Phi}$ is a map into a dot product space related to the chosen kernel function. Now, if $s(\mathbf{x}_i) < 0$, then $\mathbf{x}_i$ can be regarded as anomalous; if $s(\mathbf{x}_i) > 0$, then $\mathbf{x}_i$ can be considered normal; and if $s(\mathbf{x}_i) = 0$, then $\mathbf{x}_i$ is exactly on the boundary and its status is not determined.

k-Means Clustering

The third and final AD technique considered is a clustering technique. k-means is an unsupervised, iterative algorithm proposed by Lloyd [66]. It is one of the most popular clustering methods because of its simplicity. In k-means, M observations have to be clustered into k clusters. Each cluster is represented by the mean (centroid) of the observations it contains. The clustering is performed such that the inter-cluster similarity is minimised, while the intra-cluster similarity is maximised. The similarity is determined by the Euclidean distance of the feature value to the mean value of the observations in the cluster: the smaller the distances, the higher the similarity.

The k-means algorithm converges quickly to a local optimum. Here, $k \in \mathbb{N}$ is a hyperparameter that, for example, can be determined using the elbow method. Here, the proportion of explained variance by the model is plotted as a function of the cluster size k. For small values of k, an increasing k will explain relatively much additional variance, but less additional variance is explained when k gets large. The optimal k is the value such that there is a bend in the plot. Now, to perform AD, a cluster boundary is introduced for each of the k clusters. This is a hypersphere around the cluster mean such that 95% of all the cluster observations are within the sphere, assuming that 5% of the observations are considered anomalous. For a new observation $\mathbf{x}_i$, first the closest cluster is chosen, and then it is determined whether $\mathbf{x}_i$ is within the boundary. If it is not, it can be considered anomalous.

3.3.4 Data Transformations

Finally, we investigated whether some data transformations had a positive effect on the AD performance of the algorithms discussed in Section 3.3.3. The transformations that were considered are normalisation and standardisation. To normalise the data, the feature values were linearly scaled such that all values lie in the interval $[0, 1]$. An advantage of normalisation, or min-max scaling, is that each feature contributes equally, since all values are bounded in the same interval. Consequently, there is no feature overshadowing the other variables because of its large (absolute) values. However, a disadvantage is that the dispersion of the data is lost, possibly making it more difficult to detect anomalies. Standardisation ensures that each feature has mean 0 and variance 1. The advantage of standardisation over normalisation is that the loss of dispersion is smaller.

Since tree-based models can handle varying feature ranges, normalisation and standardisation are not required in an IF. However, the ocSVM and k-means methods are sensitive to magnitudes, and could therefore benefit from these transformations.

3.4 Experimental Setup

As mentioned in Section 3.3.3, there were several hyperparameters that had to be determined beforehand. For the IF, these constants were the sub-sampling size $\psi \in \mathbb{N}$ and the number of trees $T \in \mathbb{N}$. Liu et al. [64] argue that $\psi = 256$ and $T = 100$ are large enough to enable convergence of the average path length of each observation. Next, the parameter ν in the ocSVM was chosen to be 0.05, since we assumed that about 5% of the observations were anomalous. The number of clusters k, which is a hyperparameter for k-means, was determined by the elbow method and varied for the different experiments that were performed: $k = 2$ for no modifications, $k = 9$ for normalisation, and $k = 38$ for standardisation. Since there are no labels, there was no ground truth in the data to which the hyperparameters could be optimised.

All described procedures were performed in Python 3.6 with the libraries numPy and Pandas. The results were obtained from an evaluation set. This set was determined by the anomaly scores calculated by the three discussed ML methods. For each technique, the anomaly scores of the bookings were ranked in descending order. Then, a random subset of 10 bookings was taken from the top 30, one from the 30 scores around the median, and one from the 30 lowest scores, yielding a sample of 30 bookings for each AD method. The sample observations from the top 30 were predicted to be fraudulent, while the other observations were considered normal. In total, there were 3 samples of 30 bookings each. There was an overlap between the samples, i.e., the algorithms ranked some observations in the same regions. Hence, they were selected more than once for the samples. There were 75 unique bookings in the total of 90. The sample bookings were classified by the domain experts as fraudulent (1) or normal

Table 3.4: Results with no data transformation on method-specific evaluation samples

Model	Precision	Recall	F_1 score	F_2 score
IF	0.75	0.8	0.774	0.789
ocSVM	0.769	0.588	0.667	0.617
***k*-means**	1	0.444	0.615	0.5

(0), making it possible to assess the detection power of the algorithms. In the samples 39 fraudulent bookings were found, while 36 bookings were deemed normal.

3.5 Results

3.5.1 Performance of Anomaly Detection Techniques

To determine the quality of the models, the precision, recall, F_1 score, and F_2 score were calculated. The latter two are given by the formula

$$F_\beta = \frac{(1+\beta^2)\cdot \text{precision}\cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}},$$

where $\beta = 1$ for the F_1 score and $\beta = 2$ for the F_2 score. The F_2 score weighs recall more than the F_1 score does, i.e., it puts more emphasis on false negatives (FNs) than on false positives (FPs). This was done for the unmodified final dataset, which is the data without normalisation or standardisation. The performance metrics for the method-specific samples of size 30 are shown in Table 3.4. The construction of these samples is explained in Section 3.4. The IF performed slightly better in finding policy violations (recall $= 0.8$) than making the distinction between normal and fraudulent observations (precision $= 0.75$). This was the other way around for the ocSVM and k-means clustering, since both techniques had a higher precision than recall. Both the F_1 and F_2 scores suggest that the IF performed the best. In fraud detection, reducing FNs is usually more important than reducing FPs, since missing a fraudulent observation is deemed more harmful than raising a false alarm. FPs only bother domain experts with extra investigation time, while FNs result in a potentially large loss of revenue. Hence, the F_2 score better represents how desirably the AD method performed. Note that this evaluation was done on the different method-specific samples. Although there is some overlap between them, a direct comparison of the performance measures is risky.

The three ML methods were also applied to the complete evaluation sample. This set is the combination of the three method-specific samples of 30 observations each. The complete sample consists of 75 unique bookings. The results for the data with no modifications are presented in Table 3.5. The performance of the methods is comparable to the results on the method-specific samples in

Table 3.5: Results with no data transformation on complete evaluation sample

Model	Precision	Recall	F_1 score	F_2 score
IF	0.706	0.615	0.657	0.632
ocSVM	0.75	0.462	0.571	0.5
***k*-means**	0.8	0.308	0.444	0.351

Table 3.6: Results on transformed method-specific evaluation data

	Normalised			
Model	**Precision**	**Recall**	F_1	F_2
IF	0	0	0	0
ocSVM	0	0	0	0
***k*-means**	0.909	0.556	0.690	0.602
	Standardised			
Model	**Precision**	**Recall**	F_1	F_2
IF	1	0.067	0.125	0.082
ocSVM	0	0	0	0
***k*-means**	0.8	0.444	0.571	0.488

terms of F_1 and F_2 score. The IF still ranks the best ($F_2 = 0.632$), followed by the ocSVM ($F_2 = 0.5$) and k-means clustering ($F_2 = 0.351$). This comparison was based on the same sample for each technique, thus strengthening the claims. Since there are 39 actual fraudulent bookings and 36 normal instances, the F_2 score is expected to be approximately 0.503 when the predicted labels are assigned by unbiased coin flips. This means only the IF performed better than this threshold value.

The normalisation and standardisation procedures had remarkable influences

Table 3.7: Results on transformed complete evaluation data

	Normalised			
Model	**Precision**	**Recall**	F_1	F_2
IF	1	0.026	0.05	0.032
ocSVM	0.5	0.026	0.049	0.032
***k*-means**	0.742	0.590	0.657	0.615
	Standardised			
Model	**Precision**	**Recall**	F_1	F_2
IF	0.5	0.077	0.133	0.093
ocSVM	1	0.026	0.05	0.032
***k*-means**	0.786	0.564	0.657	0.598

Table 3.8: List of features to identify suspicious activity

List of features that detect suspicious activity
Order ratio
PoC circumvention ratio
Number of cancellations
Number of booking class switches
Number of OTA owners which are unequal to the creator

on the results, as can be seen in Tables 3.6 and 3.7. For the method-specific samples, the precision and recall are both 0 for the IF and ocSVM, performing severely worse than without data transformations. This was expected to some extent for the IF, since the segregation of the observations is done more rapidly with large variations in the data. However, this was not expected for the ocSVM. According to literature, transforming the data should benefit an SVM. This could be due to the Gaussian radial basis function that we used in this research. Nevertheless, the performance of k-means clustering increased from $F_2 = 0.5$ to $F_2 = 0.602$ with normalisation. There was a slight decrease for standardisation from $F_2 = 0.5$ to $F_2 = 0.488$.

For the combined sample, the performance of all three considered AD methods moderately increased in terms of F_2 score compared to the method-specific samples. In short, the IF performed the best on its own sample of 30 observations without any data transformations ($F_2 = 0.789$). This was also the case for the ocSVM ($F_2 = 0.617$). k-Means clustering performed the best on the complete evaluation sample with normalised data ($F_2 = 0.615$). Note that all these values are larger than the threshold value of 0.503.

3.5.2 Feature Evaluation

The results of the AD methods and the advice of the domain experts allowed us to construct a set of features which were deemed to be the most likely to identify suspicious behaviour of an OTA. The list of the five most important features is given in Table 3.8. The first feature indicates the sum of the segment identifiers divided by the corresponding triangular number: $\binom{S+1}{2}$, where S is the number of flight segments in the booking. We expect the sum to equal the triangular number (and so the feature value to be 1), since the segments are usually labelled in ascending order from 1 to S. The order ratio feature is not equal to 1 for the booking shown in Table 3.2, because flight segments have been cancelled. The second variable is related to PoC circumvention. As discussed in Section 3.3.2, the fact that PoC circumvention has occurred could indicate fraudulent behaviour. We also showed in Tables 3.1 and 3.2 how the number of cancellations, which is the third most important feature, could be connected to fraud. The fourth feature has not been discussed in the feature analysis, but an unexpected value of this feature also suggests malicious behaviour. Finally, the last feature in Table 3.8 indicates whether the OTA creating the booking is not equal to the OTA owning it.

3.6 Discussion

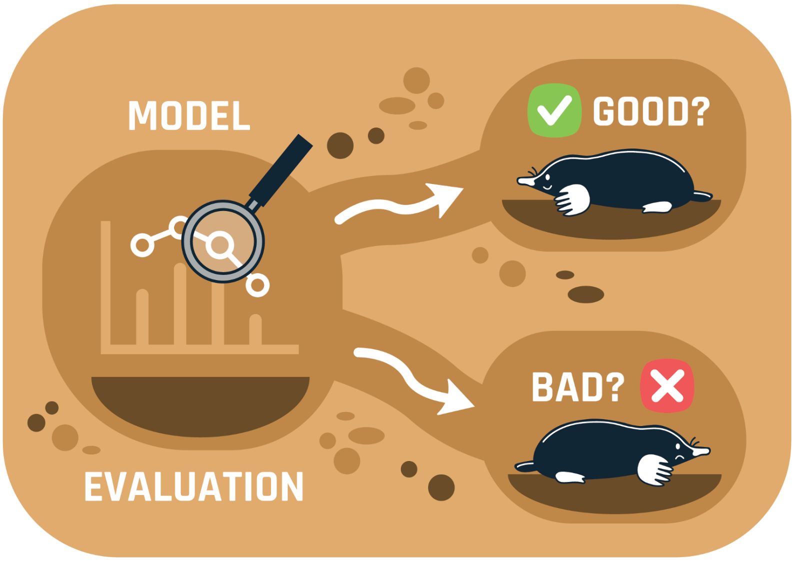
The goal of this research was to discover policy violations conducted by OTAs with the use of three AD methods. To this end, the raw data was analysed and new variables were constructed to better describe the behaviour of the OTAs. We demonstrated that these new features were important in detecting fraudulent bookings. This encourages domain experts to monitor these variables to detect some of the fraudulent behaviour and avoid revenue loss. Moreover, this advises an airline organisation to update its policy agreement with the OTAs to prevent such malpractices from happening in the future.

Together with the domain experts, we concluded that most of the anomalies were caused by cancellation activity in the bookings, suggesting that the values of the features corresponding to this behaviour give a strong indication of fraud. However, there were instances in which normal bookings were detected as fraudulent, which happened because of complex and highly unusual flights. Also, there were instances in which the domain experts marked a booking as fraudulent, but it was based on a gut feeling. Here, the benefit of using UL becomes evident: these bookings never would have been found when the bookings were only analysed on a feature-based level. Moreover, since we were not able to find literature about this research field, we took an important step in understanding fraudulent behaviour conducted by OTAs.

One of our suggestions for future research is to broaden the scope to make the results more generalisable. Firstly, we considered the bookings of one airline market. It is possible that the behaviour of OTAs is significantly different for another market. Secondly, because of time constraints, only 75 records ($\approx 0.42\%$) were analysed by the domain experts. This means the results could be notably different when a new sample is considered. Another suggestion for future research is to find out at which stage in the booking process the models are able to detect fraud in an online setting.

Part II

Model Evaluation



4 Estimating the F_1 Score for Learning from Positive and Unlabelled Examples

Het is zwaar hoor, maar wij lopen niet te zeiken.

Ellie Lust
Wie is de Mol? Renaissance

Semi-Supervised Learning can be applied to datasets that contain both labelled and unlabelled instances. It can result in more accurate predictions compared to fully supervised or unsupervised learning, in case limited labelled data is available. Positive-Unlabelled Learning (PUL) focuses on cases in which the labelled instances are only positive. Given the lack of negatively labelled data, estimating the performance is generally difficult. In this chapter, we propose a new approach to approximate the F_1 score for PUL. It requires an estimate of the fraction of all positive instances that are labelled as such. We derive theoretical properties of the approach, and apply it to several datasets to study its empirical behaviour and to compare it to the most well-known score in the field: the LL score. Results show that even when the estimate is quite off compared to the real fraction of positive labels the approximation of the F_1 score is significantly better than the LL score.

Based on [109]:
Seyed Amin Tabatabaei, Jan Klein, and Mark Hoogendoorn
Estimating the F_1 Score for Learning from Positive and Unlabeled Examples
2020 International Conference on Machine Learning, Optimization, and Data Science

4.1 Introduction

There has been a keen interest in algorithms that can learn a good classifier by using both labelled and unlabelled data. The field addressing such data is called Semi-Supervised Learning (SeSL). SeSL algorithms exploit the labelled data just like Supervised Learning (SL) algorithms do, but to improve learning they take the structure seen in the unlabelled data into account. Based on this combination, the algorithms are able to surpass the performance of fully supervised and unsupervised algorithms on partially labelled data, as Liu [62] for example show.

One category of problems in SeSL focuses on learning from datasets that only have positively labelled and unlabelled data, referred to as Positive-Unlabelled Learning (PUL). PUL is seen in multiple application domains (see [22, 110, 134]). The F_1 score is generally a prominent metric in classification problems, because considering both the precision and recall is desirable. However, in PUL, it is impossible to directly compute the F_1 score, since there are no negatively labelled examples available. Attempts to mitigate this problem have been proposed, though. For example, Lee and Liu [55] introduce the LL score. This metric shows approximately the same behaviour as the F_1 score without the need to have negatively labelled examples. However, in absolute terms, it can be quite off from the real F_1 score.

In this chapter, we present a novel approach to estimate the F_1 score for the PUL case. This estimator assumes that the fraction of labelled cases compared to the total number of positive samples is known. This assumption is not unrealistic in many cases and we show that even when the estimated fraction is somewhat off, the proposed estimator still performs better than the popular LL score. We specify the approach and perform a mathematical analysis whereby we determine the sensitivity of our approach to mistakes in the estimation of the fraction of positive labels. On top of that, we conduct a number of experiments, both using generated and real life data. We compare the estimates of both the LL score and our newly introduced metric, and we show that the estimates using our approach are: (1) significantly closer to the true F_1 score, and (2) better at selecting the 'best' model out of a set of models.

The rest of this chapter is organised as follows. The formal problem description is given in Section 4.2. Related work is presented in Section 4.3, while our proposed approach is introduced in Section 4.4 together with the mathematical analysis. The experimental setup and accompanying results are described in Sections 4.5 and 4.6, respectively. Finally, Section 4.7 concludes the chapter.

4.2 Problem Formulation

In PUL, there are instances $i \in \mathcal{M}$ that are specified by their feature vector $\mathbf{x}_i \in \mathbb{R}^K$, corresponding label $y_i \in \{-1, 1\}$, and the availability of the label $s_i \in \{0, 1\}$. Here, $\mathcal{M} := \{1, \ldots, M\}$ is the set of observations and $K \in \mathbb{N}$ is the number of features. If, for an instance i, the label is available ($s_i = 1$), then it is always positive ($y_i = 1$). If the label is not available ($s_i = 0$), it can be either positive or negative. More specifically, let $\mathcal{P} \subseteq \mathcal{M}$ be the set of observations with a positive label. Let $\mathcal{S} \subseteq \mathcal{P}$ be the subset of observations for which the positive label is provided. Consequently, the labels of the observations in $\mathcal{U} := \mathcal{M} \backslash \mathcal{S}$ are not known.

An important assumption in most PUL algorithms is that positive labelled instances are *Selected Completely At Random* among positive examples (SCAR assumption). This assumption lies at the heart of most PUL algorithms [6]. Hence, $\mathcal{S}$ is a random subset of $\mathcal{P}$ under SCAR.

Using $\mathcal{S}$ and $\mathcal{U}$, we want to build a classifier f that can predict the label of the cases in $\mathcal{U}$, i.e., ideally $f(\mathbf{x}_i) = y_i$ for $i \in \mathcal{U}$. It should be stressed that, during the training and validation process, the response values of instances outside $\mathcal{S}$ are not available. Therefore, learning should be done based on $\mathcal{S}$ combined with properties from the unlabelled data $\mathcal{U}$.

4.3 Related Work

In this section, we briefly introduce the commonly used two-step strategy to provide an intuition of PUL algorithms. This is followed by metrics that estimate the performance of the resulting models.

4.3.1 PUL Algorithms: Two-step Strategy

A well-known class of PUL algorithms is the two-step strategy [62]. In step 1, a set of reliable negative instances $\mathcal{N}_R$ is chosen from the unlabelled instances $\mathcal{U}$. In step 2, the algorithm iteratively adds more instances to $\mathcal{N}_R$, which are used as negative examples in the next iterations. This procedure is repeated until a convergence criterion is met or when no more instances are added to $\mathcal{N}_R$. There are several techniques for each of these steps. For example, the spy technique [63] and the Ricchio technique [59] are used for the first step. The Expectation-Maximization algorithm [21] can be a natural choice for the second step. A deeper review about two-step techniques is provided by Liu [62].

4.3.2 Performance Estimation

To select the classifier with the best generalisable performance, evaluation is needed. In normal SL, the F_1 score is a common performance measurement for binary classifiers. It is expressed as follows:

$$F_1 = 2 \cdot \frac{\text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}}, \tag{4.1}$$

with

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\sum_{i \in \mathcal{M}} \mathbf{1}_{\{f(\mathbf{x}_i)=1, y_i=1\}}(i)}{\sum_{i \in \mathcal{M}} \mathbf{1}_{\{y_i=1\}}(i)} = \frac{P_1}{P}, \tag{4.2}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{P_1}{\sum_{i \in \mathcal{M}} \mathbf{1}_{\{f(\mathbf{x}_i)=1\}}(i)} = \frac{P_1}{M_1}. \tag{4.3}$$

Here, $\mathbf{1}_{\{\cdot\}}$ represents the indicator function. Moreover, $P := |\mathcal{P}|$ is the number of positive instances and P_1 is the number of positive instances which are also predicted to be 1, i.e., the number of true positives (TP). Note that P is equal to TP plus the number of false negatives (FN). M_1 is the total number of observations which are predicted as positive. This is equal to TP plus the number of false positives (FP).

In PUL, the target label y_i is not available for unlabelled instances (with $s_i = 0$). Therefore, calculating the recall (Equation (4.2)) and the precision (Equation (4.3)) is not possible. However, under the SCAR assumption, we expect the fraction of predicted positives in $\mathcal{S}$ to be the same as the fraction of predicted positives in $\mathcal{P}$:

$$\mathbf{E}\left(\frac{S_1}{S}\right) \overset{SCAR}{=} \frac{P_1}{P}, \tag{4.4}$$

with S_1 the number of predicted positives in $\mathcal{S}$ and $S := |\mathcal{S}|$. Because of SCAR, the behaviour of the classifier on $\mathcal{S}$ represents its behaviour on $\mathcal{P}$. Hence, the recall can be estimated by

$$\overline{\text{rec}} = \frac{\sum_{i \in \mathcal{S}} \mathbf{1}_{\{f(\mathbf{x}_i)=1\}}(i)}{S} = \frac{S_1}{S}. \tag{4.5}$$

However, it is difficult to approximate the value of the precision, because it is less straightforward to obtain an estimate of P_1/M_1. This also means it is hard to estimate the F_1 score in PUL. To solve this, multiple approaches exist, of which the LL score is commonly used. This score is given by

$$\text{LL} = \frac{\overline{\text{rec}}^2}{M_1/M} = \frac{S_1^2 \cdot M}{S^2 \cdot M_1}.$$

It can be directly calculated from a validation set that contains positive and unlabelled examples. Moreover,

$$\frac{\text{recall}^2}{M_1/M} = \frac{P_1^2 \cdot M}{P^2 \cdot M_1} = \frac{(P_1/M_1) \cdot (P_1/P)}{P/M} = \frac{\text{precision} \cdot \text{recall}}{P/M}.$$

Therefore, the LL score also has an estimation of the precision in its definition. Lee and Liu [55] claim that the LL score has roughly the same behaviour as the F_1 score: a high value of the LL score means both precision and recall are high, while a low value means that either recall or precision is low.

4.4 Methodology

In this section, we present our approach to estimate the F_1 score in a PUL problem. It is based on the assumption that we have an approximation of the fraction of positive instances that are labelled. Moreover, we analyse our approach mathematically.

4.4.1 Approach to Estimate F_1 Score

First, we show that the fraction ρ, defined as $\rho := S/P$, supports in estimating the precision. Note that ρ represents the probability that a positive observation is labelled. Under SCAR, given in Equation (4.4), we have

$$\mathbf{E}\left(\frac{S_1}{\rho}\right) = P \cdot \mathbf{E}\left(\frac{S_1}{S}\right) \overset{SCAR}{=} P_1.$$

This allows us to estimate Equation (4.3) by

$$\overline{\text{prec}} = \frac{S_1/\rho}{M_1} = \frac{S_1}{\rho \cdot M_1}. \tag{4.6}$$

The F_1 score, given in Equation (4.1), can now be estimated by

$$\overline{F_1} := 2 \cdot \frac{\overline{\text{rec}} \cdot \overline{\text{prec}}}{\overline{\text{rec}} + \overline{\text{prec}}} = 2 \cdot \frac{(S_1/S) \cdot (S_1/(\rho \cdot M_1))}{(S_1/S) + (S_1/(\rho \cdot M_1))} = 2 \cdot \frac{S_1}{\rho \cdot M_1 + S}, \tag{4.7}$$

while the actual F_1 score can be reformulated as

$$F_1 = 2 \cdot \frac{\rho \cdot P_1}{\rho \cdot M_1 + S}.$$

We are interested in how the approximated $\overline{F_1}$ differs from the actual F_1 score. Hence, we define the variable ΔF_1 as:

$$\Delta F_1 := \overline{F_1} - F_1 = 2 \cdot \frac{S_1 - \rho \cdot P_1}{\rho \cdot M_1 + S}.$$

The actual F_1 score is fixed given the dataset and trained classifier f, but $\overline{F_1}$ depends on which subset of $\mathcal{P}$ happens to be labelled. The number of predicted positive observations S_1 has a hypergeometric distribution (see [102]) with P the population size, P_1 the number of 'success states' in the population, and S the number of draws. Here, an observation is 'successful' if it is a true positive. Thus, $S_1 \sim \text{Hypergeometric}(P, P_1, S)$. Then,

$$\begin{aligned}
\mathbf{E}(S_1) &= S \cdot \frac{P_1}{P} = \rho \cdot P_1, \\
\mathbf{Var}(S_1) &= S \cdot \frac{P_1(P - P_1)(P - S)}{P^2(P-1)} = \frac{\rho(1-\rho)P_1(S - \rho \cdot P_1)}{S - \rho}.
\end{aligned}$$

We have for the approximated recall, precision, and F_1 score:

$$\begin{aligned}
\mathbf{E}(\overline{\text{rec}}) &= \frac{\rho \cdot P_1}{S} = \text{recall}, & \mathbf{Var}(\overline{\text{rec}}) &= \frac{\mathbf{Var}(S_1)}{S^2},\\
\mathbf{E}(\overline{\text{prec}}) &= \frac{\rho \cdot P_1}{\rho \cdot M_1} = \text{precision}, & \mathbf{Var}(\overline{\text{prec}}) &= \frac{\mathbf{Var}(S_1)}{\rho^2 \cdot M_1^2},\\
\mathbf{E}(\overline{F_1}) &= 2 \cdot \frac{\rho \cdot P_1}{\rho \cdot M_1 + S} = F_1, & \mathbf{Var}(\overline{F_1}) &= \frac{4 \cdot \mathbf{Var}(S_1)}{(\rho \cdot M_1 + S)^2}.
\end{aligned}$$

Since the expected values of the estimators are equal to the actual performance metrics, the estimators are unbiased.

4.4.2 Estimating ρ

Since $\rho = S/P$, and the size S of $\mathcal{S}$ is given, estimating P means estimating ρ. There are different ways to estimate this value. To this end, domain knowledge or prior experiences with similar datasets could be exploited, or a classifier could be used to make this estimate. In the remainder of this chapter we use the classifier-based approach following Elkan and Noto [26] and we evaluate how well it works for real life cases.

4.4.3 Behaviour under Stochastic ρ

To clarify the consequences of substituting ρ by an estimator, we analyse the theoretical implications of a stochastic ρ on the estimators of the recall (Equation (4.5)), precision (Equation (4.6)), and F_1 score (Equation (4.7)). Hence, we do not know the real value of $\rho \in (0, 1]$, and it is therefore indicated by the random variable $\overline{\rho}$. Since our estimator of the recall does not involve ρ, the distribution of $\overline{\text{rec}}$ remains the same when ρ is replaced by $\overline{\rho}$. However, the estimator of the precision does change:

$$\overline{\text{prec}}_{\overline{\rho}} = \frac{S_1}{\overline{\rho} \cdot M_1}.$$

Consequently,

$$\begin{aligned}
\mathbf{E}(\overline{\text{prec}}_{\overline{\rho}}) &= \frac{\mathbf{E}(S_1)}{M_1} \cdot \mathbf{E}\left(\frac{1}{\overline{\rho}}\right) = \frac{\rho \cdot P_1}{M_1} \cdot \mathbf{E}\left(\frac{1}{\overline{\rho}}\right)\\
\mathbf{Var}(\overline{\text{prec}}_{\overline{\rho}}) &= \frac{1}{M_1^2}\left[\mathbf{E}(S_1^2) \cdot \mathbf{E}\left(\frac{1}{\overline{\rho}^2}\right) - \mathbf{E}(S_1)^2 \cdot \mathbf{E}\left(\frac{1}{\overline{\rho}}\right)^2\right]\\
&= \frac{\mathbf{Var}(S_1) \cdot \mathbf{E}\left(\frac{1}{\overline{\rho}^2}\right) + \rho^2 P_1^2 \cdot \mathbf{Var}\left(\frac{1}{\overline{\rho}}\right)}{M_1^2}.
\end{aligned}$$

This means $\overline{\text{prec}}_{\overline{\rho}}$ is an unbiased estimator only when $\mathbf{E}(1/\overline{\rho}) = 1/\rho$, which is not true in general. More specifically, consider the convex function $\varphi : (0, 1] \to [1, \infty)$ given by $\varphi(x) = 1/x$. Hence, by Jensen's inequality, $\varphi(\mathbf{E}(X)) \leq$

$\mathbf{E}(\varphi(X))$ for random variable X and convex function φ. Thus, $1/\rho \leq \mathbf{E}(1/\overline{\rho})$, and so

$$\mathbf{E}(\overline{\text{prec}_{\overline{\rho}}}) = \frac{\rho P_1}{M_1} \cdot \mathbf{E}\left(\frac{1}{\overline{\rho}}\right) \geq \frac{P_1}{M_1} = \text{precision}.$$

The approximated F_1 score with stochastic ρ is given by

$$\overline{F_1}_{\overline{\rho}} := 2 \cdot \frac{S_1}{\overline{\rho} \cdot M_1 + S}.$$

The expected value of this estimator is at least equal to the actual F_1 score, which means it is biased. We show this again using Jensen's inequality and the convex function $\varphi : (0, 1] \to (\frac{1}{M_1+S}, \frac{1}{S}]$ given by $\varphi(x) = \frac{1}{M_1 \cdot x + S}$. Now,

$$\begin{aligned}\mathbf{E}(\overline{F_1}_{\overline{\rho}}) &= 2\mathbf{E}(S_1) \cdot \mathbf{E}\left(\frac{1}{\overline{\rho} \cdot M_1 + S}\right)\\ &\geq 2\mathbf{E}(S_1) \cdot \frac{1}{\mathbf{E}(\overline{\rho}) \cdot M_1 + S} = 2 \cdot \rho \cdot P_1 \frac{1}{\rho \cdot M_1 + S} = F_1.\end{aligned}$$

Consequently, when the fraction of labelled observations among the positive instances is deemed stochastic with an arbitrary distribution, then both the estimators of the precision and F_1 score are expected to overestimate.

4.5 Experimental Setup

In order to evaluate our approach we empirically compared it to the real F_1 score and to the behaviour of the LL score on four different datasets. For these datasets the ground truth for all instances is available.

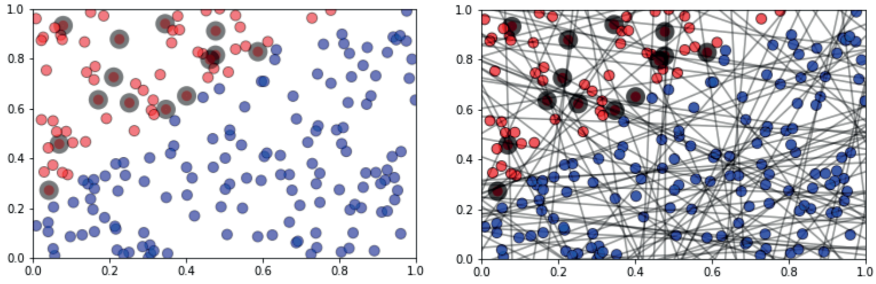
4.5.1 Datasets and Setup

Generated Dataset

The first dataset, or actually collection of datasets, was generated randomly. These datasets contain two features $X_1, X_2 \in [0, 1]$ and points were generated uniformly random. In order to assign the points to one of the two classes ($y = 0$ or $y = 1$), their position was compared to a randomly generated line ($X_2 = X_1 + 0.2$). When the observation is above the line, it was considered positive, otherwise it was negative. We then selected a random sample (SCAR assumption) of size $S := \rho \cdot P$ of the positive examples ($y = 1$) to act as $\mathcal{S}$ (with value $s = 1$) and we took the rest as $\mathcal{U}$ (with $s = 0$). Figure 4.1a shows a randomly generated dataset, and Figure 4.1b shows randomly generated linear classifiers.

Iris Dataset

The Iris Dataset is popular in pattern recognition and Machine Learning literature [24]. It contains 3 flower classes (*setosa*, *versicolor*, and *virginica*) of 50



(a) Example of generated dataset **(b)** Randomly generated linear classifiers

Figure 4.1: Example of a generated dataset with accompanying classifiers. Positive labels are red and negatives are blue. Labelled samples are highlighted with a grey circle

instances each. There are 4 features available for each instance. By taking the last class (*virginica*) as positive and the two others as negative, it transfers to a binary classification problem. These two classes are not linearly separable. We again made a random fraction ρ of all positively labelled data points available as $\mathcal{S}$, the rest being $\mathcal{U}$. 4-D linear hyper planes were generated randomly to act as classifiers, similar to what was done for the Generated Dataset.

Heart Disease Dataset

The Heart Disease Dataset is well-known in pattern recognition literature [24]. The data contains both numerical and categorical features. The goal of models applied to this dataset is to predict the presence of heart disease in a patient. We trained random forest models with different numbers of estimators (randomly chosen between 1 and 100) and maximum depth (randomly selected between 1 and 10).

Health Dataset

Finally, the Health Dataset was obtained from the VU University Medical Center and contains event logs of more than 300,000 patients [110]. The goal is to identify certain types of patients based on their event log. Part of these patients are labelled as suffering from kidney disease, others are labelled as having diabetes, and the rest have another disease. For each disease, a fraction ρ of positive examples were randomly selected as labelled examples $\mathcal{S}$, while the rest were taken as unlabelled examples $\mathcal{U}$. Following [110], two features are present in the dataset to predict the label, namely $X_1, X_2 \in \mathbb{Z}$, that summarise the care paths of patients in a way that patients with that disease are optimally separable. A classifier was defined by a set of two thresholds, (θ_1, θ_2). An instance was predicted as positive if $X_1 > \theta_1$ and $X_2 > \theta_2$.

4.5.2 Experimental Conditions and Performance Metrics

We compared our approach to the LL score based on two metrics: *(i)* distance to the real F_1 score; and *(ii)* percentage of inversions. We computed the Root Mean Square Error (RMSE) to measure the distance to the F_1 score. Computing the percentage of inversions, which is the key in showing that the right model was selected and thus our most important outcome, was a bit more difficult. The inversions were used to show how often the wrong model was selected based on either $\overline{F_1}$ or the LL score compared to the actual F_1 score. To this end, we took the different classifiers for each dataset and compared them pairwise. We have an *inversion* each time a classifier that has a higher F_1 score compared to the other classifier is ranked lower. Hence, we wanted to minimise the number of inversions. We compared the results using a Wilcoxon paired test to show possibly significant differences [120].

We conducted three types of experiments: *(i)* empirically studying the assumptions and theoretical results of our approach; *(ii)* evaluating the performance of our approach with the true value of ρ being available; and *(iii)* evaluating the performance with stochastic ρ. Each is explained in more detail below.

Empirical Evaluation of Assumptions

As has become clear, we made the assumption that ρ can be estimated. We have presented various approaches to estimate ρ, one of which involves a classifier g. This estimator is exactly correct if $g(\mathbf{x}) = \Pr(s = 1|\mathbf{x})$ for all $\mathbf{x}$, but usually this condition does not hold in practice. To show the applicability of this technique (and how easily we can obtain the crucial ρ) we used the Iris Dataset and the Generated Dataset with different values of ρ, and estimated the value of ρ using a trained classifier on the labelled data points. We conducted this experiment 100 times per value of ρ. To get more accurate results, we used the one-leave-out cross-validation technique.

Secondly, we evaluated another part of our approach, namely our result on the bounds. In this experiment, we again used the Generated Dataset and Iris Dataset, and took one randomly selected linear classifier with a real F_1 score of 0.44. We then took different values for ρ and for each value drew a random sample 100 times, thereby estimating the F_1 score using our approach. We used these results to compute the mean estimated value and the confidence bounds. We compared these to the bound following our mathematical result.

Performance Evaluation with Known ρ

To evaluate the approach compared to the LL score, we first assume ρ to be known and correct. For these experiments, we selected the value of ρ ranging from 0 to 1 with increments of 0.01. For each setting of ρ for the Generated Dataset we generated 200 datasets and 100 random lines to act as classifiers. For the Iris Dataset and the Health Dataset we generated 100 random classifiers. We measured the performance for both the deviation from the F_1 score

Table 4.1: Datasets used for the various experiments

Experiment	**Generated**	**Iris**	**Heart**	**Health**
Estimating ρ	✓	✓		
Evaluating bounds	✓	✓		
Perf. Eval. Known ρ: F_1	✓	✓	✓	✓
Perf. Eval. Known ρ: Inversions	✓	✓	✓	✓
Perf. Eval. Stoch. ρ: Inversions	✓	✓		

and number of inversions.

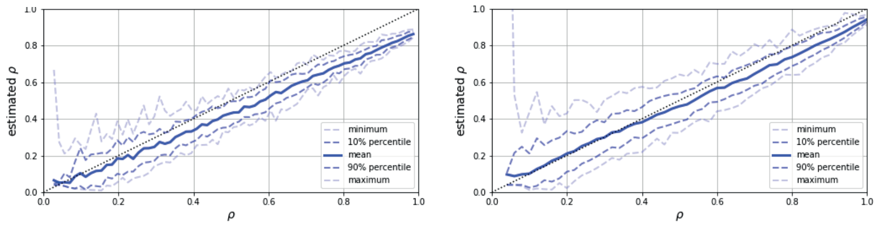
Performance Evaluation with Stochastic ρ

For stochastic ρ, we varied the noise level and use the same experimental setup as presented under the *known ρ case*. The noise level was varied from a 50% underestimation to a 100% overestimation. Due to the computational complexity, we only studied this part on the Generated Dataset and Iris Dataset and measured the percentage of inversions. Table 4.1 gives a brief overview of the datasets used for the various experiments.

4.6 Results

First, we report the results of the empirical evaluation of the assumptions followed by experiments in which the correct value of ρ was known. Then, we explore the cases where the value of ρ was noisy (either under- or overestimated).

4.6.1 Checking the Assumptions



(a) Generated Dataset **(b)** Iris Dataset

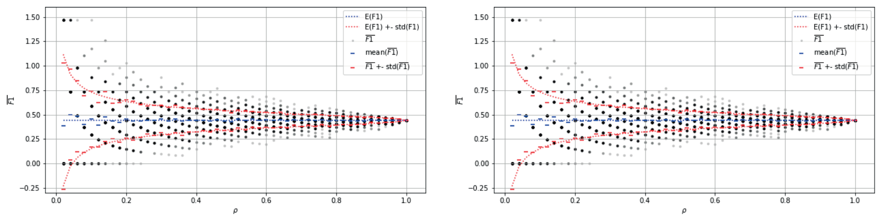
Figure 4.2: Estimating ρ using a classifier (see Elkan and Noto [26]). For each value of ρ, this experiment is conducted 50 times. Mean, minimum, maximum, and 10th and 90th percentiles are reported

Figure 4.2 shows the results on the estimation of ρ through our classifier including confidence bounds. We see that as ρ increases the variability of the

Table 4.2: RMSE of real F_1 vs. $\overline{F_1}$, and F_1 vs. LL score. For all cases, $\rho = 0.30$

	Generated	**Iris**	**Heart Disease**	**Health**
F_1	0.064	0.060	0.089	0.060
LL-score	0.772	0.420	0.344	0.623

estimation decreases, which makes sense as a small sample will make the estimation very sensitive to the sample drawn. However, the plots show that the estimations are very reasonable. We do not observe any obvious difference in the estimation behaviour between the Generated Dataset (Figure 4.2a) and the Iris Dataset (Figure 4.2b).



(a) Generated Dataset **(b)** Iris Dataset

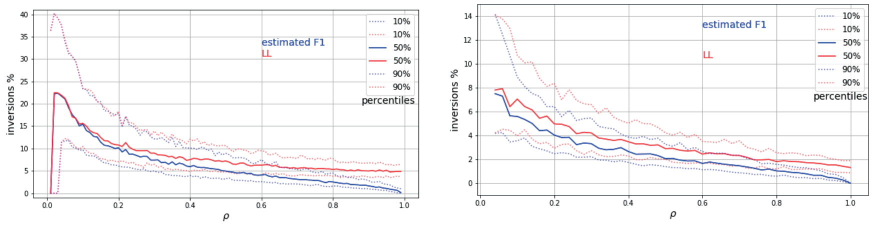
Figure 4.3: Expected value and standard deviation of estimated $\overline{F_1}$ for different values of ρ. Each grey point shows $\overline{F_1}$ for one set of labelled data points. Points which overlap become darker. The empirical mean and bounds of $\overline{F_1}$ are shown by the blue and red dashed line, respectively, while the mean and bounds computed based on our mathematical result are shown by blue and red dashed curves

Our second study about the underlying assumptions concerns our estimation of the bounds. Figure 4.3 shows the empirical results for various values of ρ, the empirical mean and bounds, and the computed mean and bounds based on our mathematical results for both the Generated Dataset and Iris Datasets. Results show that the two align very well for both data configurations.

4.6.2 Known ρ

Let us move on to measuring the performance of our approach. We start by considering the case in which ρ was equal to the true value. Table 4.2 reports the RMSE for different datasets. The RMSE for our approach is much smaller compared to the LL score. This was also to be expected as the proposed score is an estimation of the F_1 score, while the LL score aims to only approximate its behaviour and not necessarily its actual value. Most important to observe (as our aim is model selection and hyperparameter optimisation) is that our estimated values are monotonically increasing with the true F_1 score. Therefore, our central metric is the number of inversions when performing model selection. Figure 4.4a shows the results for the Generated Dataset for varying values of ρ.

We see that as ρ increases the difference in performance between our approach and the LL score increases in favour of the approach we put forward. Also, the confidence intervals become smaller as ρ increases. Moreover, we see that our approach never performs worse. Results of a paired Wilcoxon signed-rank test show that for values of $\rho > 0.05$ the number of inversions caused by sorting classifiers based on the $\overline{F_1}$ score is significantly lower than those by the LL score. Moving on to the Iris Dataset, Figure 4.4b shows the average number of inversions for different values of ρ. Our approach is significantly better when $\rho > 0.02$. For the Heart Disease Dataset, the Wilcoxon paired test shows a significant better performance for our approach if $\rho > 0.02$. Finally, for the real-life *Health Dataset* our approach is significantly better when $\rho > 0.08$ for kidney disorder and $\rho > 0.02$ for diabetes.



(a) Generated Dataset **(b)** Iris Dataset

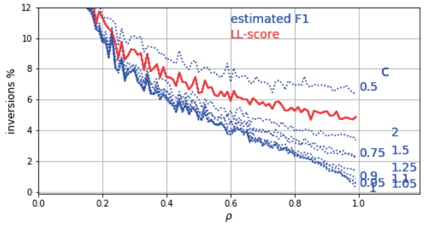
Figure 4.4: Number of inversions for both the LL score (red) and the proposed $\overline{F_1}$ (blue) including confidence bounds for different values of ρ

4.6.3 Stochastic ρ

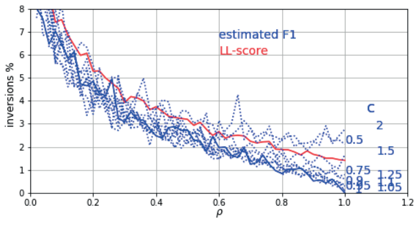
In many cases, we do not know the exact value of ρ and might only be able to estimate it (see our first set of experiments). Figure 4.5 shows how under- and overestimations influence the number of inversions of our proposed approach for the Generated Dataset and the Iris Dataset. Here, the true value of ρ is multiplied with a value c. When considering the Generated Dataset, we see that only for a value of $c = 0.5$, i.e., an extreme underestimation of ρ, the proposed approach scores worse compared to the LL score. For the Iris Dataset, we see a similar pattern, except that for a value of $c = 2$, i.e., a severe overestimation, our performance is worse. This shows that suffering from a bit of noise does not hamper our approach.

4.7 Discussion

In this chapter we have introduced a novel way of estimating the F_1 score to enable model selection and hyperparameter tuning in PUL. This novel method is based on the assumption that an estimation can be made on the fraction of labelled positive cases. A mathematical analysis was performed to show the



(a) Generated Dataset



(b) Iris Dataset

Figure 4.5: Effect of error in estimated ρ on the percentage of inversions

expected value of the estimation with respect to the real F_1 score. Also, we analysed what the influence of stochasticity in ρ is on the estimations. We showed that the estimators become biased when ρ is stochastic, while they are unbiased when there is no noise.

Furthermore, we conducted experiments to evaluate our assumptions empirically, showing that the approach is practically applicable. On top, we have empirically compared our proposed approach to a well-known metric for model selection, namely the LL score. Results show that our approach (1) is closer to the true F_1 score, and (2) has fewer wrong selections of models (i.e., inversions) compared to the LL score for a variety of datasets. Both cases only hold for sufficiently large samples of training data, though the approach never performs worse. When considering wrongly estimating the fraction of positive labels we see that only severe under- and overestimations hamper performance compared to the LL score. Our approach also brings advantages that the whole family of F_β scores can now be estimated for any $\beta > 0$.

Suggestions for future work are, firstly, to develop a different estimator of the precision. The estimator $\overline{\text{prec}}$ was given by $S_1/(\rho M_1)$. This measure takes values in $[0, 1/\rho]$ depending on S_1 and M_1 (which are determined by the classifier). Consequently, it is possible for the estimated precision to be larger than 1 which always leads to an overestimation of the actual precision, since the latter is by definition bounded in $[0, 1]$. This also results in an estimator of the F_1 score which can exceed 1. Hence, a different estimator of the precision could be constructed such that it is bounded in $[0, 1]$, but still expected to be equal to the actual precision. This could also solve or diminish the problem of the expected overestimation of the precision and F_1 score when ρ is noisy. Secondly, we want to explore whether the performance reports generalises over other datasets and also other types of models that are compared.

5 The Dutch Draw: Constructing a Universal Baseline for Binary Prediction Models

Ik vind het fijn als alles in harmonie is.

Loretta Schrijver
Wie is de Mol? 2010

Novel prediction methods should always be compared to a baseline to know how well they perform. Without this frame of reference, the performance score of a model is basically meaningless. What does it mean when a model achieves an accuracy of 0.8 on a test set? This depends on the dataset. Therefore, a proper baseline is needed to evaluate the "goodness" of a performance score. Comparing with the latest state-of-the-art model is usually insightful. However, this can change rapidly when newer models are developed. Also, reproducibility is problem when models become more complex. This chapter presents a standardised baseline for all binary classification models, named the Dutch Draw. It provides mathematical properties for many commonly used evaluation metrics. The Dutch Draw baseline is: (1) *general*, as it is applicable to all binary classification problems; (2) *simple*, as it is quickly determined without training or parameter-tuning; and (3) *informative*, as insightful conclusions can be drawn from the results. Moreover, the Dutch Draw baseline is more sophisticated than any dummy classifier. The Dutch Draw baseline is therefore ideal for two purposes. Firstly, as a sanity check during the development process of a model. Secondly, as a way to not only use the current state-of-the-art model, but to additionally use this robust and universal baseline to enable comparisons across research domains. We showed that in a considerable number of studies the performance of learning methods was worse than the Dutch Draw baseline. This emphasises the necessity for the use of a universal baseline.

Based on [118]:
Etienne van de Bijl[1], *Jan Klein*[1], *Joris Pries*[1], *Sandjai Bhulai, Mark Hoogendoorn, and Rob van der Mei*
The Dutch Draw: Constructing a Universal Baseline for Binary Prediction Models
(Submitted for publication)

[1]contributed equally

5.1 Introduction

A typical data science project can be crudely simplified to the following steps: *(i)* understanding the problem context, *(ii)* comprehending the data, *(iii)* preparing the data, *(iv)* modelling, *(v)* evaluating the model, and *(vi)* deploying the model [122]. Before deploying a new model, it should be tested whether it meets certain predefined success criteria. Such an assessment should indicate whether the model performs as desired and if it outperforms other models based on the relevant evaluation metrics. A baseline is a crucial and integral part in this evaluation, as it gives an indication of the actual performance of a model.

However, what baseline should be selected? Of course, a good baseline is desirable, but what explicitly makes a baseline 'good'? Comparing with the latest state-of-the-art model is usually insightful. However, this can change rapidly when newer models are developed. Moreover, reproducibility of models is often a problem, because their code is not published or large amounts of computational resources are required. These aspects make it hard or even impossible to compare older results with newer research. Yet, we are not criticising the comparison of a new model with a state-of-the-art model per se. However, we are pleading for an additional standardised baseline that makes it possible to compare results across research domains and papers. Some fundamental baselines are already used for this, such as dummy classifiers or optimal threshold classifiers. However, these baselines also their weaknesses. This leads to a quest for a good standardised baseline: one baseline that rules them all. We outline three principal elements that any good standardised baseline should have: *generalness*, *simplicity*, and *informativeness*.

Generalness Most commonly in research, a new model is compared to a limited number of existing models in a specific field. Although these are usually carefully selected, they are still subjectively chosen. Take binary classification, in which the objective is to label each observation either zero or one. Here, one could already select a decision tree [73], a random forest [20], variants of naive Bayes [119], k-nearest neighbours [4], a support vector machine [90], a neural network [108], or a logistic regression model [99] to evaluate the performance. These models are often trained specifically for a problem instance with parameters tuned for optimal performance in that specific case. Hence, these methods are not general. One could not take a decision tree that is used for determining bankruptcy [73] and use it as a baseline for a pathological voice detection problem [80]. At least structural adaptations and retraining are necessary. A good standard baseline should be applicable to all binary classification problems, irrespective of the specific domain.

Simplicity A baseline should not be complex. Although it is hard to determine whenever a baseline is too complex, two components are essential. Firstly, it is necessary for applicability that a baseline can be determined relatively fast. Training a neural network many times to generate an average baseline or optimising the parameters of a certain model could take a long time. Secondly, if a standardised baseline is very complex, it can be harder to draw meaningful conclusions. Is it expected that a new model is outperformed by this ingeniously complex baseline, or is it exactly what one would expect? This leads to the last property of a good standard baseline.

Informativeness A baseline should be informative. If a method achieves a score higher or lower than the baseline, a clear conclusion needs to be drawn. Is it obvious that the baseline should be beaten? On the other side, if this is too evident, does the baseline actually give additional information? A baseline should give as much information as possible while still being able to derive meaningful conclusions from it.

These principles express a list of requirements for baseline models. Additional requirements can be made, but the three principles should be fundamental. This chapter focuses on the construction of a universal baseline satisfying the three principles for binary classification problems. The resulting framework could then be used to develop baselines in other fields of Supervised Learning, such as multi-class classification and regression. Nevertheless, let us first introduce previously mentioned fundamental baselines and discuss to what extent they satisfy the three principles.

Dummy Classifier A dummy classifier is a non-learning model that makes predictions using a simple set of rules. For example, always predicting the most frequent class label or predicting each class with some probability. A dummy classifier is simple and general, however it is not always informative. The information gained by performing better than a simple dummy classifier can be small. With the plethora of dummy classifiers, selection is also still arbitrary and questionable.

Optimal Threshold Classifier Koyejo et al. [49] determined for a large family of binary performance measures that the optimal classifier consists of a sign function with a threshold tailored to each specific measure. To determine the optimal classifier, it is necessary to know or approximate $\mathbb{P}(y = 1|\mathbf{X} = \mathbf{x})$, which is the probability that the binary label y is 1 given the features $\mathbf{X} = \mathbf{x}$. Lipton et al. [61] had a similar approach, but they only focused on the F_1 score. These probabilities should be learned from training data. But in what way should these probabilities be learned, and will they be accurate? Again, this leads to an arbitrary selection of an approximation model. It is a clever approach, but unfortunately there is no clear-cut answer for how to approximate these conditional probabilities. If they are not accurate, the optimal classifier is based on wrong information, which makes it hard to draw meaningful conclusions from this approach.

Both the dummy classifier and the optimal threshold classifier have their strengths and weaknesses. In our approach, we combine their strengths into the *Dutch Draw*. This can be seen as a dummy classifier on steroids. Instead of arbitrarily choosing a dummy classifier, we mathematically derive which classifier, from a family of classifiers, has the best expected performance. Also, this expected performance can be directly determined, making it very fast to obtain the baseline. The Dutch Draw is: *(i)* applicable to any binary classification problem, *(ii)* reproducible, *(iii)* simple, *(iv)* parameter-free, *(v)* more informative than any dummy classifier, and *(vi)* an explainable minimal requirement for any new model. This makes the Dutch Draw an ideal candidate for a standardised baseline in binary classification.

Our contributions are as follows: *(i)* we introduce the Dutch Draw and explain why this method produces a universal baseline that is general, simple and informative for all binary classification problems; *(ii)* we provide the mathematical properties of the Dutch Draw for many evaluation metrics and summarise them in several tables; and *(iii)* we calculated the Dutch Draw baseline for several Machine Learning (ML) methods from literature and showed that in some cases our simple, non-trained baseline outperforms the methods, which strongly supports its necessity.

To provide an overview of our paper: in Section 5.2, preliminaries of binary classification are discussed and the list of the considered evaluation metrics is given. Section 5.3 discusses the Dutch Draw, its theoretical results, and specifically, the baselines. In Section 5.4, we give a visual representation of the characteristics of the Dutch Draw. In addition, the performances of many ML methods applied to commonly used binary classification problems are benchmarked by the corresponding Dutch Draw baselines. Lastly, Section 5.5 provides the conclusion of this research and further study directions.

5.2 Preliminaries

Before formulating the Dutch Draw, we need to introduce necessary notation, and simultaneously, provide elementary information on binary classification. This is required to explain how binary models are evaluated. As there are several measures to determine performance, we discuss how these measures are constructed and which are selected for this research.

5.2.1 Binary Classification

The dataset used to learn the relationship between the explanatory or input variables and the binary response or output variable consists of $M \in \mathbb{N}_{>0}$ observations. Let $\mathcal{M} := \{1, \ldots, M\}$ be the set of observation indices. Each instance, denoted by $\mathbf{x}_i$, has $K \in \mathbb{N}_{>0}$ explanatory feature values. These features can be categorical or numerical. Without loss of generality, we assume that $\mathbf{x}_i \in \mathbb{R}^K$ for all $i \in \mathcal{M}$. Moreover, each observation has a corresponding output value $y_i \in \{0, 1\}$. Now, let $\mathbf{X} := [\mathbf{x}_1 \ldots \mathbf{x}_M]^T \in \mathbb{R}^{M \times K}$ denote the matrix with all observations and their explanatory feature values, and let $\mathbf{y} = (y_1, \ldots, y_M)$ be the response vector. The complete dataset is then represented by $(\mathbf{X}, \mathbf{y})$. We call the observations with response value 1 'positive', while the observations with response value 0 are 'negative'. Let P denote the number of positives and N the number of negatives. Note that $P + N = M$.

5.2.2 Evaluation Metrics

An *evaluation metric* quantifies the prediction performance of a trained model. We categorise the evaluation metrics into two groups: *base measures* and *performance measures*. Since there are two possible values for both the predicted and the true classes in binary classification, there are four base measures: the number of true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN). Any function of one or more of these base measures can be a performance measure. To shorten notation, let $\hat{P} := \mathrm{TP} + \mathrm{FP}$ and $\hat{N} := \mathrm{TN} + \mathrm{FN}$ denote the number of positively and negatively predicted instances, respectively.

Table 5.1: ***Definitions and domains:*** Overview of the definition and the codomain of each evaluation metric considered in this research

Metric	Definition	Codomain
True Positives (TP)	TP	$\mathbb{N}_0$
False Positives (FP)	FP	$\mathbb{N}_0$
False Negatives (FN)	FN	$\mathbb{N}_0$
True Negatives (TN)	TN	$\mathbb{N}_0$
True Positive Rate (TPR), Recall, Sensitivity	$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}}$	$[0,1]$
False Positive Rate (FPR), Fall-out	$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{N}}$	$[0,1]$
False Negative Rate (FNR), Miss Rate	$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{P}}$	$[0,1]$
True Negative Rate (TNR), Specificity, Selectivity	$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}}$	$[0,1]$
Positive Predictive Value (PPV), Precision	$\mathrm{PPV} = \frac{\mathrm{TP}}{\hat{P}}$	$[0,1]$
Negative Predictive Value (NPV)	$\mathrm{NPV} = \frac{\mathrm{TN}}{\hat{N}}$	$[0,1]$
False Discovery Rate (FDR)	$\mathrm{FDR} = \frac{\mathrm{FP}}{\hat{P}}$	$[0,1]$
False Omission Rate (FOR)	$\mathrm{FOR} = \frac{\mathrm{FN}}{\hat{N}}$	$[0,1]$
F_β score (F_β)	$F_\beta = (1+\beta^2)/\left(\frac{1}{\mathrm{PPV}} + \frac{\beta^2}{\mathrm{TPR}}\right)$	$[0,1]$
Youden's J Statistic (J), (Bookmaker) Informedness	$\mathrm{J} = \mathrm{TPR} + \mathrm{TNR} - 1$	$[-1,1]$
Markedness (MK)	$\mathrm{MK} = \mathrm{PPV} + \mathrm{NPV} - 1$	$[-1,1]$
Accuracy (Acc)	$\mathrm{Acc} = \frac{\mathrm{TP}+\mathrm{TN}}{M}$	$[0,1]$
Balanced Accuracy (BAcc)	$\mathrm{BAcc} = \frac{1}{2}(\mathrm{TPR} + \mathrm{TNR})$	$[0,1]$
Matthews Correlation Coefficient (MCC)	$\mathrm{MCC} = \frac{\mathrm{TP}\cdot\mathrm{TN}-\mathrm{FP}\cdot\mathrm{FN}}{\sqrt{\hat{P}\cdot\hat{N}\cdot\mathrm{P}\cdot\mathrm{N}}}$	$[-1,1]$
Cohen's kappa (κ)	$\kappa = \frac{\mathrm{P}_o - \mathrm{P}_e}{1-\mathrm{P}_e}$, with $\mathrm{P}_o = \mathrm{Acc}, \mathrm{P}_e = \frac{\hat{P}\cdot P + \hat{N}\cdot N}{M^2}$	$[-1,1]$
Fowlkes-Mallows Index (FM), G-mean 1	$\mathrm{FM} = \sqrt{\mathrm{TPR}\cdot\mathrm{PPV}}$	$[0,1]$
G-mean 2 ($\mathrm{G}^{(2)}$)	$\mathrm{G}^{(2)} = \sqrt{\mathrm{TPR}\cdot\mathrm{TNR}}$	$[0,1]$
Prevalence Threshold (PT)	$\mathrm{PT} = \frac{\sqrt{\mathrm{TPR}\cdot\mathrm{FPR}}-\mathrm{FPR}}{\mathrm{TPR}-\mathrm{FPR}}$	$[0,1]$
Threat Score (TS), Critical Success Index	$\mathrm{TS} = \frac{\mathrm{TP}}{P+\mathrm{FP}}$	$[0,1]$

All performance measures and base measures that we consider in this research are shown in Table 5.1. Here, also their abbreviations, possibly alternative names, their definitions, and corresponding codomains are presented. These codomains show in what set the measure can theoretically take values (without considering the exact values of P, N, $\hat{P}$, and $\hat{N}$). In Section 5.3, the case-specific codomains are provided when we discuss the evaluation metrics in more detail. Finally, note that the list is not exhaustive, but it contains most of the commonly used evaluation metrics.

Not every evaluation metric is always well-defined. Most notably, there is a problem with the Prevalence Threshold (PT). As Table 5.1 shows, PT is undefined whenever its numerator and denominator are both zero. Unfortunately, this is not an exception that can easily be discarded. Therefore, we omit PT throughout the rest of this research. For more details on this, see Section 5.6.22. Furthermore, a problem for several performance measures is division by zero. For example, the *True Positive Rate* (TPR) defined as TPR $=$ TP$/P$ cannot be calculated whenever $P = 0$. Therefore, we have made assumptions for the allowed values of P, N, $\hat{P}$, and $\hat{N}$. These are shown in Table 5.2.

Table 5.2: ***Assumptions on domains P, N, $\hat{P}$, and $\hat{N}$:*** Some measures are not defined if P, N, $\hat{P}$ or $\hat{N}$ is equal to zero. The assumptions in this table are therefore necessary (always $M > 0$)

Metric	Domain P	Domain N	Domain $\hat{P}$	Domain $\hat{N}$
TP, FP, FN, TN, Acc, κ	$\mathbb{N}_0$	$\mathbb{N}_0$	$\mathbb{N}_0$	$\mathbb{N}_0$
TPR, FNR, TS	$\mathbb{N}_{>0}$	$\mathbb{N}_0$	$\mathbb{N}_0$	$\mathbb{N}_0$
FPR, TNR	$\mathbb{N}_0$	$\mathbb{N}_{>0}$	$\mathbb{N}_0$	$\mathbb{N}_0$
PPV, FDR	$\mathbb{N}_0$	$\mathbb{N}_0$	$\mathbb{N}_{>0}$	$\mathbb{N}_0$
NPV, FOR	$\mathbb{N}_0$	$\mathbb{N}_0$	$\mathbb{N}_0$	$\mathbb{N}_{>0}$
F_β, FM	$\mathbb{N}_{>0}$	$\mathbb{N}_0$	$\mathbb{N}_{>0}$	$\mathbb{N}_0$
J, BAcc, G$^{(2)}$	$\mathbb{N}_{>0}$	$\mathbb{N}_{>0}$	$\mathbb{N}_0$	$\mathbb{N}_0$
MK	$\mathbb{N}_0$	$\mathbb{N}_0$	$\mathbb{N}_{>0}$	$\mathbb{N}_{>0}$
MCC	$\mathbb{N}_{>0}$	$\mathbb{N}_{>0}$	$\mathbb{N}_{>0}$	$\mathbb{N}_{>0}$

5.3 Methodology

In this section, we introduce the Dutch Draw framework and we discuss how this method is able to provide a universal baseline for any evaluation metric. This baseline is general, simple, and informative, which are crucial for a good standardised baseline, as we explained in Section 5.1. Firstly, we provide the family of Dutch Draw classifiers, and secondly, we explain how the optimal classifier generates the baseline.

5.3.1 Dutch Draw Classifier

The goal of our research is to provide a universal baseline for any evaluation metric in binary classification. These standardised baselines are generated by the optimal Dutch Draw classifier. Before we discuss how to select the best Dutch Draw classifier, we first present the definition of the general Dutch Draw classifier. This is the function $\sigma_\theta : \mathbb{R}^{M \times K} \to \{0,1\}^M$ with as input an evaluation dataset with M observations and K feature values per observation. It generates the predictions for these observations by outputting a vector of M binary predictions. It is defined as:

$$\begin{aligned}\sigma_\theta(\mathbf{X}) :=\{&\text{take a random sample without replacement of size } \lfloor M \cdot \theta \rceil \\ &\text{of rows from } \mathbf{X} \text{ and assign 1 to these observations and 0} \\ &\text{to the remaining rows}\}.\end{aligned}$$

Here, $\lfloor \cdot \rceil$ is the function that rounds its argument to the nearest integer. The parameter $\theta \in [0,1]$ controls what percentage of observations are predicted as positive. The mathematical definition of σ_θ is given by:

$$\sigma_\theta(\mathbf{X}) := (\mathbf{1}_E(i))_{i \in \mathcal{M}} \text{ with } E \subseteq \mathcal{M} \text{ uniformly drawn s.t. } |E| = \lfloor M \cdot \theta \rceil,$$

with $(\mathbf{1}_E(i))_{i \in \mathcal{M}}$ the vector with ones in the positions in E and zeroes elsewhere. Note that σ_θ does not learn from the features in the data, just as a dummy classifier. The set of all Dutch Draw classifiers $\{\sigma_\theta : \theta \in [0,1]\}$ is the complete family of models that classify a random sample of any size as positive.

Given a Dutch Draw classifier, the number of predicted positives $\hat{P}$ depends on θ and is given by $\hat{P}_\theta := \lfloor M \cdot \theta \rceil$. The number of predicted negatives is $\hat{N}_\theta := M - \lfloor M \cdot \theta \rceil$. To be specific, these two numbers are integers, and thus, different values of θ can lead to the same value of $\hat{P}_\theta$. Therefore, we introduce the parameter $\theta^* := \frac{\lfloor M \cdot \theta \rceil}{M}$ as the discretised version of θ. Furthermore, we define

$$\Theta^* := \left\{ \frac{\lfloor M \cdot \theta \rceil}{M} : \theta \in [0,1] \right\} = \left\{ 0, \frac{1}{M}, \ldots, \frac{M-1}{M}, 1 \right\}$$

as the set of all unique values that θ^* can obtain for all $\theta \in [0,1]$.

Given a performance measure and dataset, each classifier σ_θ has an expected performance score. Optimising over all $\theta \in [0,1]$ gives the classifier with the best expected performance. This is the Dutch Draw baseline. It is general, because it is obtained from an approach that is not trained on feature values and there are no free parameters to be tuned. The baseline is simple, because it can be calculated very fast and it is clear that any learning model should outperform this baseline. The baseline is more informative than any dummy classifier, because it is by definition the best classifier out of the family of dummy classifiers.

Distributions Evaluation Metrics

The distributions of the base measures are directly determined by σ_θ. Consider for example TP: the number of positive observations that are also predicted to be positive. In a dataset of M observations with P labelled positive, $\lfloor M \cdot \theta \rceil$ random observations are predicted as positive under the Dutch Draw approach. This implies that TP_θ is hypergeometrically distributed with parameters M, P, and $\lfloor M \cdot \theta \rceil$, as the classifier randomly draws $\lfloor M \cdot \theta \rceil$ samples without replacement from a population of size M, where P samples are labelled positive. Thus:

$$\mathbb{P}(\text{TP}_\theta = s) = \begin{cases} \frac{\binom{P}{s} \cdot \binom{M-P}{\lfloor M \cdot \theta \rceil - s}}{\binom{M}{\lfloor M \cdot \theta \rceil}} & \text{if } s \in \mathcal{D}(\text{TP}_\theta), \\ 0 & \text{else,} \end{cases}$$

where $\mathcal{D}(\text{TP}_\theta)$ is the domain of TP_θ. The definition of this domain is given in Equation (5.1).

The other three base measures are also hypergeometrically distributed following similar reasoning. This leads to

$$\begin{aligned} \text{TP}_\theta &\sim \text{Hypergeometric}(M, P, \lfloor M \cdot \theta \rceil), \\ \text{FP}_\theta &\sim \text{Hypergeometric}(M, N, \lfloor M \cdot \theta \rceil), \\ \text{FN}_\theta &\sim \text{Hypergeometric}(M, P, M - \lfloor M \cdot \theta \rceil), \\ \text{TN}_\theta &\sim \text{Hypergeometric}(M, N, M - \lfloor M \cdot \theta \rceil). \end{aligned}$$

Note that these random variables are not independent. In fact, they can all be written in terms of TP_θ. This is a very nice property that follows from Dutch Draw classifiers. Furthermore, most evaluation metrics can be written as a linear combination of only TP_θ. With only one random variable, theoretical derivations and optimal classifiers can be determined. As mentioned before, $\text{TP}_\theta + \text{FN}_\theta = P$ and $\text{TN}_\theta + \text{FP}_\theta = N = M - P$, and we also have $\text{TP}_\theta + \text{FP}_\theta = \lfloor M \cdot \theta \rceil$, because this denotes the total number of positively predicted observations. These three identities are linear in TP_θ, thus each base measure can be written in the form $Z_\theta(a, b) := a \cdot \text{TP}_\theta + b$ with $a, b \in \mathbb{R}$. Let $H_\theta(a, b)$ be the probability distribution of $Z_\theta(a, b)$. Then, by combining the identities:

$$\text{TP}_\theta = \text{TP}_\theta \qquad = Z_\theta(1, 0) \qquad \sim H_\theta(1, 0), \tag{B1}$$

$$\text{FP}_\theta = \hat{P}_\theta - \text{TP}_\theta \qquad = Z_\theta\left(-1, \hat{P}_\theta\right) \qquad \sim H_\theta\left(-1, \hat{P}_\theta\right), \tag{B2}$$

$$\text{FN}_\theta = P - \text{TP}_\theta \qquad = Z_\theta(-1, P) \qquad \sim H_\theta(-1, P), \tag{B3}$$

$$\text{TN}_\theta = N - \hat{P}_\theta + \text{TP}_\theta \qquad = Z_\theta\left(1, N - \hat{P}_\theta\right) \qquad \sim H_\theta\left(1, N - \hat{P}_\theta\right), \tag{B4}$$

where $N = M - P$ and $\hat{P}_\theta := \lfloor M \cdot \theta \rceil$.

Distribution F_β Score To give an illustration of $H_\theta\left(a,b\right)$ for a performance measure, we consider the F_β *score* ($\mathrm{F}_\theta^{(\beta)}$), introduced by Chinchor [16]. It is the weighted harmonic average between the True Positive Rate (TPR_θ) and the Positive Predictive Value (PPV_θ). The latter two performance measures are discussed extensively in Sections 5.6.5 and 5.6.9, respectively. The F_β score balances predicting the actual positive observations correctly (TPR_θ) and being cautious in predicting observations as positive (PPV_θ). The factor $\beta > 0$ indicates how much more TPR_θ is weighted compared to PPV_θ. The F_β score is commonly defined as

$$\mathrm{F}_\theta^{(\beta)} = \frac{1+\beta^2}{\frac{1}{\mathrm{PPV}_\theta} + \frac{\beta^2}{\mathrm{TPR}_\theta}}.$$

$\mathrm{F}_\theta^{(\beta)}$ can be formulated in terms of only TP_θ by using the definitions of PPV_θ and TPR_θ given in Table 5.1 and Equations (B1) and (B2):

$$\mathrm{F}_\theta^{(\beta)} = \frac{(1+\beta^2)\mathrm{TP}_\theta}{\beta^2\cdot P + \lfloor M\cdot\theta\rceil}.$$

Since PPV_θ is only defined when $\lfloor M\cdot\theta\rceil > 0$ and TPR_θ only when $P > 0$, we need for $\mathrm{F}_\theta^{(\beta)}$ that both these restrictions hold. The definition of $\mathrm{F}_\theta^{(\beta)}$ is linear in TP_θ and can therefore be formulated as

$$\mathrm{F}_\theta^{(\beta)} = Z_\theta\left(\frac{1+\beta^2}{\beta^2\cdot P + \lfloor M\cdot\theta\rceil}, 0\right),$$

Ranges Evaluation Metrics

The values that $Z_\theta\left(a,b\right)$ can attain depend on a and b, and of course, on the domain of TP_θ. Without restriction, the maximum number that TP_θ can be is P. Then, all positive observations are correctly predicted. However, when θ is small enough such that $\lfloor M\cdot\theta\rceil < P$, then only $\lfloor M\cdot\theta\rceil$ observations are predicted as positive. Consequently, TP_θ can only reach the value $\lfloor M\cdot\theta\rceil$. Hence, in general, the upper bound on the domain of TP_θ is $\min\{P, \lfloor M\cdot\theta\rceil\}$. The same reasoning holds for the lower bound: when θ is small enough, the minimum number of TP_θ is 0, since all positive observations can be incorrectly predicted. However, when θ gets large enough, positive observations have to be predicted positive even if all $M-P$ negative observations are predicted positive. Thus, in general, the lower bound on the domain is $\max\{0, \lfloor M\cdot\theta\rceil - (M-P)\}$. Now, let $\mathcal{D}(\mathrm{TP}_\theta)$ be the domain of TP_θ, then:

$$\mathcal{D}(\mathrm{TP}_\theta) := \{i\in\mathbb{N}_0 : \max\{0, \lfloor M\cdot\theta\rceil - (M-P)\} \le i \le \min\{P, \lfloor M\cdot\theta\rceil\}\}. \tag{5.1}$$

Consequently, the range of $Z_\theta\left(a,b\right)$ is given by

$$\mathcal{R}\left(Z_\theta\left(a,b\right)\right) := \{a\cdot i + b\}_{i\in\mathcal{D}(\mathrm{TP}_\theta)}. \tag{R}$$

Expectations Evaluation Metrics

The introduction of $Z_\theta\,(a,b)$ allows us to write its expected value in terms of a and b. This statistic is important, as it is required to calculate the actual baseline. Since TP_θ has a Hypergeometric$(M, P, \lfloor M\cdot\theta\rceil)$ distribution, its expected value is known and given by

$$\mathbf{E}[\mathrm{TP}_\theta] = \frac{\lfloor M\cdot\theta\rceil}{M}\cdot P.$$

Next, we obtain the following general definition for the expectation of $Z_\theta\,(a,b)$:

$$\mathbf{E}[Z_\theta\,(a,b)] = a\cdot\mathbf{E}[\mathrm{TP}_\theta] + b = a\cdot\frac{\lfloor M\cdot\theta\rceil}{M}\cdot P + b, \tag{E}$$

This rule is consistently used to determine the expectation for each measure.

Expectation $\mathbf{F}_\theta^{(\beta)}$ To illustrate, we look at $\mathrm{F}_\theta^{(\beta)}$. The F_β score is linear in TP_θ with slope $a = (1+\beta^2)/(\beta^2\cdot P + \lfloor M\cdot\theta\rceil)$ and intercept $b = 0$, and so, its expectation is given by:

$$\begin{aligned}\mathbf{E}[\mathrm{F}_\theta^{(\beta)}] = \mathbf{E}\left[Z_\theta\left(\frac{1+\beta^2}{\beta^2\cdot P+\lfloor M\cdot\theta\rceil}, 0\right)\right] &\overset{(E)}{=} \frac{1+\beta^2}{\beta^2\cdot P+\lfloor M\cdot\theta\rceil}\cdot\mathbf{E}[\mathrm{TP}_\theta] + 0\\ &= \frac{\lfloor M\cdot\theta\rceil\cdot P\cdot(1+\beta^2)}{M\cdot(\beta^2\cdot P+\lfloor M\cdot\theta\rceil)}\\ &= \frac{(1+\beta^2)\cdot P\cdot\theta^*}{\beta^2\cdot P + M\cdot\theta^*}.\end{aligned} \tag{5.2}$$

A full overview of the distribution and mean of all considered base and performance measures is given in Table 5.3. All the calculations performed to derive the corresponding distributions and expectations are provided in Section 5.6.

5.3.2 Optimal Dutch Draw Classifier and Baselines

Next, we use the general results that we obtained in Section 5.3.1 to determine the universal Dutch Draw baseline for each evaluation metric. More specifically, the expectation as a function of θ is used for this. Given an evaluation metric, the standardised baseline is obtained by optimising (taking the minimum or maximum of) the associated expectation over $\theta\in[0,1]$. Hence, we are interested in the optimal value θ_{opt} for θ, and the corresponding Dutch Draw classifier $\sigma_{\theta_{\text{opt}}}$. To select the best classifier, it is crucial to discuss what knowledge is available to the classifier. This can directly influence the optimality of a classifier. We consider three cases:

1. knowledge of P and M (full knowledge)

Table 5.3: ***Properties metrics under Dutch Draw:*** The mean score is optimised over all allowed $\theta^* \in \Theta^*$ for each measure to obtain the Dutch Draw baseline. "✗" denotes that no closed-form expression was found

Metric	Expectation	Distribution $H_\theta(a, b)$	
		a	b
TP	$\theta^* \cdot P$	1	0
FP	$\theta^*(M-P)$	-1	$M \cdot \theta^*$
FN	$(1-\theta^*)P$	-1	P
TN	$(1-\theta^*)(M-P)$	1	$M - P - M \cdot \theta^*$
TPR	θ^*	$\frac{1}{P}$	0
FPR	θ^*	$-\frac{1}{M-P}$	$\frac{M \cdot \theta^*}{M-P}$
FNR	$1-\theta^*$	$-\frac{1}{P}$	1
TNR	$1-\theta^*$	$\frac{1}{M-P}$	$1 - \frac{M \cdot \theta^*}{M-P}$
PPV	$\frac{P}{M}$	$\frac{1}{M \cdot \theta^*}$	0
NPV	$1-\frac{P}{M}$	$\frac{1}{M(1-\theta^*)}$	$1 - \frac{P}{M(1-\theta^*)}$
FDR	$1-\frac{P}{M}$	$-\frac{1}{M \cdot \theta^*}$	1
FOR	$\frac{P}{M}$	$-\frac{1}{M(1-\theta^*)}$	$\frac{P}{M(1-\theta^*)}$
F_β	$\frac{(1+\beta^2)\theta^* \cdot P}{\beta^2 \cdot P + M \cdot \theta^*}$	$\frac{1+\beta^2}{\beta^2 \cdot P + M \cdot \theta^*}$	0
J	0	$\frac{M}{P(M-P)}$	$-\frac{M \cdot \theta^*}{M-P}$
MK	0	$\frac{1}{M \cdot \theta^*(1-\theta^*)}$	$-\frac{P}{M(1-\theta^*)}$
Acc	$\frac{(1-\theta^*)(M-P)+\theta^* \cdot P}{M}$	$\frac{2}{M}$	$1 - \theta^* - \frac{P}{M}$
BAcc	$\frac{1}{2}$	$\frac{M}{2P(M-P)}$	$\frac{1}{2} - \frac{M \cdot \theta^*}{2(M-P)}$
MCC	0	$\frac{1}{\sqrt{P(M-P)\theta^*(1-\theta^*)}}$	$-\frac{\sqrt{P \cdot \theta^*}}{\sqrt{(M-P)(1-\theta^*)}}$
κ	0	$\frac{2}{P(1-\theta^*)+(M-P)\theta^*}$	$-\frac{2\theta^* \cdot P}{P(1-\theta^*)+(M-P)\theta^*}$
FM	$\sqrt{\frac{\theta^* \cdot P}{M}}$	$\frac{1}{\sqrt{P \cdot M \cdot \theta^*}}$	0
$\mathrm{G}^{(2)}$	✗	Non-linear in TP_θ	Non-linear in TP_θ
TS	✗	Non-linear in TP_θ	Non-linear in TP_θ

2. knowledge of only M
3. no knowledge.

A classifier is considered to be optimal if the expectation is optimal given the available knowledge. In the first case, the total number of positives P and negatives $M - P$ is known. Note that information about individual samples is not assumed. Although it is only a small bit of information, this is often not available to other classification models. Thus, we also discuss the second and third case. In an offline setting, the total number of samples M is often known. This only tells a classification model how often it should classify and does not give any information about the labels of the samples. In an online setting, M is unknown, as there is a stream of samples with undefined length that needs to be classified by a model. The complete Dutch Draw framework with the three cases is shown in Figure 5.1.

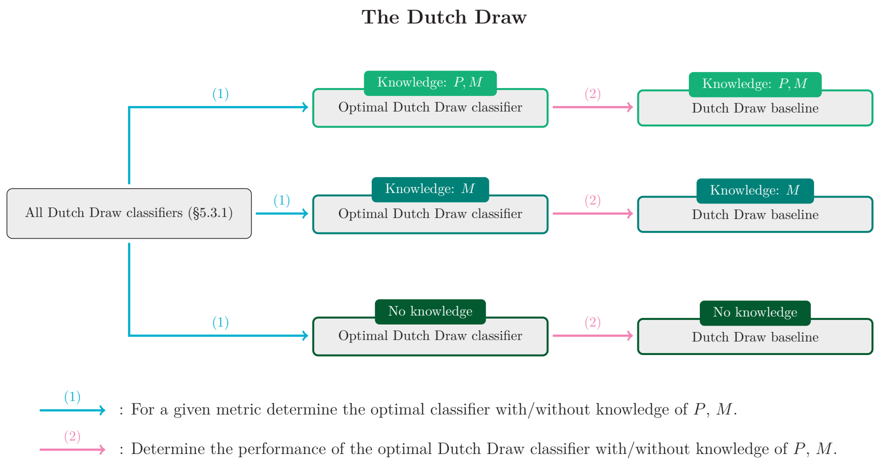

The Dutch Draw
All Dutch Draw classifiers (§5.3.1)
(1)
Knowledge: P, M
Optimal Dutch Draw classifier
(2)
Knowledge: P, M
Dutch Draw baseline
Knowledge: M
Optimal Dutch Draw classifier
Knowledge: M
Dutch Draw baseline
No knowledge
Optimal Dutch Draw classifier
No knowledge
Dutch Draw baseline
(1) : For a given metric determine the optimal classifier with/without knowledge of P, M.
(2) : Determine the performance of the optimal Dutch Draw classifier with/without knowledge of P, M.

Figure 5.1: ***Framework of the Dutch Draw:*** There are three cases of knowledge, leading to three different pathways. For each case, the classifier that results in the optimal expectation, and thus the baseline, is selected from the group of all Dutch Draw classifiers

Dutch Draw Baselines with Knowledge of P and M

In this section, we provide the optimal expectations, and therefore the baselines, and the corresponding optimisers θ_{opt} for the evaluation metrics when P and M are known. Here, we explicitly show the calculations for $\text{F}_{\theta}^{(\beta)}$ to give an illustration of how the baseline is determined for an evaluation metric that is linear in TP_{θ}. All derived universal baselines and corresponding optimisers are shown in Table 5.4.

Optimising $\mathbf{E}[F_\beta]$ To determine the extreme values of the expectation of $\mathrm{F}_\theta^{(\beta)}$, and therefore the baselines, the derivative of the function $f : [0,1] \to [0,1]$ defined as

$$f(t) = \frac{(1+\beta^2)\cdot P \cdot t}{\beta^2 \cdot P + M \cdot t}$$

is calculated. First note that $f(\lfloor M \cdot \theta \rceil / M) = \mathbf{E}[\mathrm{F}_\theta^{(\beta)}]$ for the expectation as defined in Equation (5.2). The derivative of f is given by

$$\frac{\mathrm{d}f(t)}{\mathrm{d}t} = \frac{\beta^2(1+\beta^2)\cdot P^2}{(\beta^2 \cdot P + M \cdot t)^2}.$$

This is strictly positive for all t in its domain, thus f is strictly increasing in t. This means $\mathbf{E}[\mathrm{F}_\theta^{(\beta)}]$ is non-decreasing in θ and also in θ^*, because the term $\theta^* = \lfloor M \cdot \theta \rceil / M$ is non-decreasing in θ. Hence, the extreme values of the expectation of $\mathrm{F}_\theta^{(\beta)}$ are its border values:

$$\begin{aligned}
\min_{\theta \in [1/(2M),1]} \left(\mathbf{E}[\mathrm{F}_\theta^{(\beta)}]\right) &= \min_{\theta \in [1/(2M),1]} \left(\frac{(1+\beta^2)\cdot P \cdot \lfloor M \cdot \theta \rceil}{M \cdot (\beta^2 \cdot P + \lfloor M \cdot \theta \rceil)}\right) \\
&= \frac{(1+\beta^2)\cdot P}{M(\beta^2 \cdot P + 1)}, \\
\max_{\theta \in [1/(2M),1]} \left(\mathbf{E}[\mathrm{F}_\theta^{(\beta)}]\right) &= \max_{\theta \in [1/(2M),1]} \left(\frac{(1+\beta^2)\cdot P \cdot \lfloor M \cdot \theta \rceil}{M \cdot (\beta^2 \cdot P + \lfloor M \cdot \theta \rceil)}\right) \\
&= \frac{(1+\beta^2)\cdot P}{\beta^2 \cdot P + M}.
\end{aligned}$$

Note that $\lfloor M \cdot \theta \rceil > 0$ is a restriction for $\mathrm{F}_\theta^{(\beta)}$, and hence the optima are taken over the interval $[1/(2M), 1]$. Furthermore, the optimisation values $\theta_{\min}$ and $\theta_{\max}$ for the extreme values are given by

$$\begin{aligned}
\theta_{\min} \in \operatorname*{arg\,min}_{\theta \in [1/(2M),1]} \left(\mathbf{E}[\mathrm{F}_\theta^{(\beta)}]\right) &= \operatorname*{arg\,min}_{\theta \in [1/(2M),1]} \left(\frac{\lfloor M \cdot \theta \rceil}{\beta^2 \cdot P + \lfloor M \cdot \theta \rceil}\right) \\
= \begin{cases} [\frac{1}{2}, 1] & \text{if } M = 1 \\ \left[\frac{1}{2M}, \frac{3}{2M}\right) & \text{if } M > 1, \end{cases}
\end{aligned}$$

$$\begin{aligned}
\theta_{\max} \in \operatorname*{arg\,max}_{\theta \in [1/(2M),1]} \left(\mathbf{E}[\mathrm{F}_\theta^{(\beta)}]\right) &= \operatorname*{arg\,max}_{\theta \in [1/(2M),1]} \left(\frac{\lfloor M \cdot \theta \rceil}{\beta^2 \cdot P + \lfloor M \cdot \theta \rceil}\right) \\
= \left[1 - \frac{1}{2M}, 1\right],
\end{aligned}$$

respectively. Following this reasoning, the discrete forms θ^*_{min} and θ^*_{max} are given by

$$\theta^*_{\text{min}} \in \underset{\theta^* \in \Theta^* \setminus \{0\}}{\arg\min} \left\{ \mathbf{E}[\mathrm{F}^{(\beta)}_{\theta^*}] \right\} = \underset{\theta^* \in \Theta^* \setminus \{0\}}{\arg\min} \left\{ \frac{\theta^*}{\beta^2 \cdot P + M \cdot \theta^*} \right\} = \left\{ \frac{1}{M} \right\},$$
$$\theta^*_{\text{max}} \in \underset{\theta^* \in \Theta^* \setminus \{0\}}{\arg\max} \left\{ \mathbf{E}[\mathrm{F}^{(\beta)}_{\theta^*}] \right\} = \underset{\theta^* \in \Theta^* \setminus \{0\}}{\arg\max} \left\{ \frac{\theta^*}{\beta^2 \cdot P + M \cdot \theta^*} \right\} = \{1\}.$$

The smallest expected $\mathrm{F}^{(\beta)}_{\theta}$ is obtained when all evaluation observations but one are predicted negative, while predicting everything positive yields the largest expected $\mathrm{F}^{(\beta)}_{\theta}$.

Dutch Draw Baselines with Knowledge Only of M

We now assume that only M is known, as P is often not available for other prediction models. This is the second branch of Figure 5.1. An important observation from Table 5.4 is that the Accuracy (Acc_θ) is the only metric for which the optimal parameter is dependent on P. This means that knowing P is not necessary for all other measures. Except possibly for $\mathrm{G}^{(2)}_{\theta}$, as no optimal threshold was derived. Thus, not knowing P does not change the optimal threshold for most measures, but how does it change the optimal threshold for Acc_θ?

Optimising Accuracy The derivation of the expectation of Acc_θ is given in Section 5.6.16. It is defined as

$$\mathbf{E}[\mathrm{Acc}_\theta | P] = \frac{(M - \lfloor M \cdot \theta \rceil)(M - P) + \lfloor M \cdot \theta \rceil \cdot P}{M^2}.$$

If test and training set have similar distributions, P can be estimated by the total number of positives in the training set. Note that it is only necessary to determine whether $P \geq \frac{M}{2}$ or vice versa. This means its explicit value is not necessary. In a less perfect world, the training set could have distinctly different distribution of labels compared to the test set.

Assume that P has distribution G_M for a given M. It then holds that

$$\begin{aligned}\mathbf{E}_{P\sim G_M}[\mathbf{E}[\mathrm{Acc}_\theta \mid P]] &= \mathbf{E}_{P\sim G_M}\left[\frac{(M - \lfloor M \cdot \theta \rceil)(M - P) + \lfloor M \cdot \theta \rceil \cdot P}{M^2}\right] \\ &= \frac{(M - \lfloor M \cdot \theta \rceil)(M - \mathbf{E}_P[P]) + \lfloor M \cdot \theta \rceil \cdot \mathbf{E}_P[P]}{M^2}.\end{aligned}$$

Equivalent to what we show in Section 5.6.16, the equation above is non-decreasing in θ when $\mathbf{E}_{P\sim G_M}[P] \geq \frac{M}{2}$ and otherwise non-increasing. This means the optimisation values θ_{min} and θ_{max} (Equations (5.20) and (5.21), respectively) are the same but P is replaced by its expectation $\mathbf{E}_{P\sim G_M}[P]$.

Table 5.4: ***Mathematical Results Dutch Draw:*** For each evaluation metric, the minimum and maximum of the expectation over all allowed $\theta^* \in \Theta^*$ lead to the relevant Dutch Draw baselines. The values of θ^* leading to these optima are the minimisers and maximisers, respectively. "✗" denotes that no closed-form expression was found

Metric	$\max\{\mathbf{E}\}$	$\boldsymbol{\Theta}^{\star}_{\text{max}} := \arg\max\{\mathbf{E}\}$	$\min\{\mathbf{E}\}$	$\boldsymbol{\Theta}^{\star}_{\text{min}} := \arg\min\{\mathbf{E}\}$
TP	P	$\{1\}$	0	$\{0\}$
FP	$M-P$	$\{1\}$	0	$\{0\}$
FN	P	$\{0\}$	0	$\{1\}$
TN	$M-P$	$\{0\}$	0	$\{1\}$
TPR	1	$\{1\}$	0	$\{0\}$
FPR	1	$\{1\}$	0	$\{0\}$
FNR	1	$\{0\}$	0	$\{1\}$
TNR	1	$\{0\}$	0	$\{1\}$
PPV	$\frac{P}{M}$	$\Theta^* \setminus \{0\}$	$\frac{P}{M}$	$\Theta^* \setminus \{0\}$
NPV	$1-\frac{P}{M}$	$\Theta^* \setminus \{1\}$	$1-\frac{P}{M}$	$\Theta^* \setminus \{1\}$
FDR	$1-\frac{P}{M}$	$\Theta^* \setminus \{0\}$	$1-\frac{P}{M}$	$\Theta^* \setminus \{0\}$
FOR	$\frac{P}{M}$	$\Theta^* \setminus \{1\}$	$\frac{P}{M}$	$\Theta^* \setminus \{1\}$
F_β	$\frac{(1+\beta^2)\cdot P}{\beta^2\cdot P+M}$	$\{1\}$	$\frac{(1+\beta^2)\cdot P}{M(\beta^2\cdot P+1)}$	$\left\{\frac{1}{M}\right\}$
J	0	Θ^*	0	Θ^*
MK	0	$\Theta^* \setminus \{0,1\}$	0	$\Theta^* \setminus \{0,1\}$
Acc	$\max\left\{\frac{P}{M}, 1-\frac{P}{M}\right\}$	Equation (5.23)	$\min\left\{\frac{P}{M}, 1-\frac{P}{M}\right\}$	Equation (5.22)
BAcc	$\frac{1}{2}$	Θ^*	$\frac{1}{2}$	Θ^*
MCC	0	$\Theta^* \setminus \{0,1\}$	0	$\Theta^* \setminus \{0,1\}$
κ	0	Θ^* ($\Theta^* \setminus \{1\}$ if $P=M$)	0	Θ^* ($\Theta^* \setminus \{1\}$ if $P=M$)
FM	$\sqrt{\frac{P}{M}}$	$\{1\}$	$\frac{\sqrt{P}}{M}$	$\left\{\frac{1}{M}\right\}$
$\text{G}^{(2)}$	✗	✗	0	$\{0,1\}$
TS	$\frac{P}{M}$	$\{1\}$	0	$\{0\}$

Hence,

$$\theta_{\min} \in \underset{\theta \in [0,1]}{\arg\min} \left(\mathbf{E}[\mathrm{Acc}_\theta]\right) = \begin{cases} \left[1 - \frac{1}{2M}, 1\right] & \text{if } \mathbf{E}_{P \sim G_M}[P] < \frac{M}{2} \\ [0,1] & \text{if } \mathbf{E}_{P \sim G_M}[P] = \frac{M}{2} \\ \left[0, \frac{1}{2M}\right) & \text{if } \mathbf{E}_{P \sim G_M}[P] > \frac{M}{2}, \end{cases}$$

$$\theta_{\max} \in \underset{\theta \in [0,1]}{\arg\max} \left(\mathbf{E}[\mathrm{Acc}_\theta]\right) = \begin{cases} \left[0, \frac{1}{2M}\right) & \text{if } \mathbf{E}_{P \sim G_M}[P] < \frac{M}{2} \\ [0,1] & \text{if } \mathbf{E}_{P \sim G_M}[P] = \frac{M}{2} \\ \left[1 - \frac{1}{2M}, 1\right] & \text{if } \mathbf{E}_{P \sim G_M}[P] > \frac{M}{2}. \end{cases}$$

Consequently, the discrete versions $\theta^*_{\min} \in \Theta^*$ and $\theta^*_{\max} \in \Theta^*$ of the optimisers are given by:

$$\theta^*_{\min} \in \underset{\theta^* \in \Theta^*}{\arg\min} \left\{\mathbf{E}[\mathrm{Acc}_{\theta^*}]\right\} = \begin{cases} \{1\} & \text{if } \mathbf{E}_{P \sim G_M}[P] < \frac{M}{2} \\ \Theta^* & \text{if } \mathbf{E}_{P \sim G_M}[P] = \frac{M}{2} \\ \{0\} & \text{if } \mathbf{E}_{P \sim G_M}[P] > \frac{M}{2}, \end{cases} \tag{5.3}$$

$$\theta^*_{\max} \in \underset{\theta^* \in \Theta^*}{\arg\max} \left\{\mathbf{E}[\mathrm{Acc}_{\theta^*}]\right\} = \begin{cases} \{0\} & \text{if } \mathbf{E}_{P \sim G_M}[P] < \frac{M}{2} \\ \Theta^* & \text{if } \mathbf{E}_{P \sim G_M}[P] = \frac{M}{2} \\ \{1\} & \text{if } \mathbf{E}_{P \sim G_M}[P] > \frac{M}{2}, \end{cases} \tag{5.4}$$

respectively. This means that we only have to know whether we expect a majority or a minority of positive observations in the evaluation set to determine the optimal optimisers. We do not have to know the precise value of $\mathbf{E}_{P \sim G_M}[P]$ or the distribution G_M of P. This is a loose assumption, as in most situations it is known whether positive observations are the majority or minority due to the nature of the data. However, what if even this is unknown?

Assume no information about G_M is known. Therefore, we want to find the classifier that leads to the maximal and minimal performance in their respective worst-case scenarios. G_M can be any distribution, thus for the maximal performance we must evaluate the worst possible distribution with respect to the chosen θ. In other words, the worst-case scenario minimises the expected Acc_θ over P and $N = M - P$ for a given θ. For the minimal performance, the worst-case scenario maximises the expected Acc_θ over P and N for given θ.

Maximal Performance in Worst-case Scenario The expected Acc_θ written in terms of P and N is obtained from Table 5.3:

$$\frac{(1-\theta^*) \cdot N + \theta^* \cdot P}{P + N}. \tag{5.5}$$

If $\theta^* \leq \frac{1}{2}$, most observations are labelled negative. Thus, $N = 0$ is the worst-case scenario for these values of θ^*. Vice versa, if $\theta^* \geq \frac{1}{2}$, most instances are labelled positive. Thus, $P = 0$ is the worst-case scenario for these values of θ^*.

The following then holds for the worst-case scenario

$$\begin{aligned}
&\min_{P,N\in\mathbb{N}:P+N>0}\left\{\frac{(1-\theta^*)\cdot N+\theta^*\cdot P}{P+N}\right\}\\
&\quad=\begin{cases}\min_{P\in\mathbb{N}_{>0}}\left\{\frac{(1-\theta^*)\cdot 0+\theta^*\cdot P}{P+0}\right\} & \text{if } \theta^*\leq\frac{1}{2},\\ \min_{N\in\mathbb{N}_{>0}}\left\{\frac{(1-\theta^*)\cdot N+\theta^*\cdot 0}{0+N}\right\} & \text{if } \theta^*\geq\frac{1}{2}.\end{cases}\\
&\quad=\min\left\{\theta^*,1-\theta^*\right\}.
\end{aligned}\tag{5.6}$$

Next, we want to find the thresholds θ^* that maximise the expected Acc_θ in the worst-case scenario. Using Equation (5.6) it follows that

$$\begin{aligned}
&\operatorname*{arg\,max}_{\theta^*\in\Theta^*}\left\{\min_{P,N\in\mathbb{N}:P+N>0}\left\{\frac{(1-\theta^*)\cdot N+\theta^*\cdot P}{P+N}\right\}\right\}\\
&\quad=\operatorname*{arg\,max}_{\theta^*\in\Theta^*}\left\{\min\left\{\theta^*,1-\theta^*\right\}\right\}\\
&\quad=\begin{cases}\left\{\frac{1}{2}\right\} & \text{if } M \text{ is even,}\\ \left\{\frac{M-1}{2M},\frac{M+1}{2M}\right\} & \text{if } M \text{ is odd.}\end{cases}
\end{aligned}\tag{5.7}$$

Thus, if no information about G_M is known, the expected Acc_θ is maximised in the worst-case scenario when $\theta^*=\frac{1}{2}$ if M is even, or $\theta^*\in\left\{\frac{M-1}{2M},\frac{M+1}{2M}\right\}$ when M is odd.

Minimal Performance in Worst-case Scenario The expected Acc_θ in terms of P and N is given in Equation (5.5). If $\theta^*\leq\frac{1}{2}$, most observations are labelled negative. Thus, $P=0$ is the worst-case scenario, because we want to obtain a small Accuracy. If $\theta^*\geq\frac{1}{2}$, then most observations are predicted positive, and thus, $N=0$ is the worst-case scenario for these θ^*. The reasoning is similar to what we derived in Equation (5.6). Hence,

$$\begin{aligned}
&\max_{P,N\in\mathbb{N}:P+N>0}\left\{\frac{(1-\theta^*)\cdot N+\theta^*\cdot P}{P+N}\right\}\\
&\quad=\begin{cases}\max_{N\in\mathbb{N}_{>0}}\left\{\frac{(1-\theta^*)\cdot N+\theta^*\cdot 0}{0+N}\right\} & \text{if } \theta^*\leq\frac{1}{2},\\ \max_{P\in\mathbb{N}_{>0}}\left\{\frac{(1-\theta^*)\cdot 0+\theta^*\cdot P}{P+0}\right\} & \text{if } \theta^*\geq\frac{1}{2}.\end{cases}\\
&\quad=\max\left\{\theta^*,1-\theta^*\right\}.
\end{aligned}\tag{5.8}$$

Now, we want to find the values θ^* such that Acc_θ is minimised in the worst-case scenario. By using Equation (5.8), it follows that

$$\begin{aligned}
&\operatorname*{arg\,min}_{\theta^*\in\Theta^*}\left\{\max_{P,N\in\mathbb{N}:P+N>0}\left\{\frac{(1-\theta^*)\cdot N+\theta^*\cdot P}{P+N}\right\}\right\}\\
&\quad=\operatorname*{arg\,min}_{\theta^*\in\Theta^*}\left\{\max\left\{\theta^*,1-\theta^*\right\}\right\}\\
&\quad=\{0,1\}.
\end{aligned}$$

Dutch Draw Baselines with No Knowledge

Finally, we consider the possibility that the total number of evaluation observations M is not known, which is the third branch of Figure 5.1. This situation can occur in for example an online setting, in which a stream of samples with undefined length should be classified. Note that any Dutch Draw classifier inherently uses M, as it labels a random subset of some size as positive. To take a random subset, M must be known. Yet, for many performance measures the optimal parameter values are either zero or one, which removes the dependency on the knowledge of M. In most other cases, the expected baseline is constant, making it also unnecessary to know M. However, a problem does arise for FM_θ and $\mathrm{F}_\theta^{(\beta)}$. Table 5.4 shows that $\Theta^\star_{\min} = \{\frac{1}{M}\}$ for these performance measures. As M is unknown, it is impossible to select the corresponding optimal classifier. If it is assumed that the order of the samples is random, an alternative optimal classifier can be defined. $\theta^*_{\min} = \frac{1}{M}$ means that only one sample is classified as 1. Now, let the first observed sample be classified as 1 and classify all other samples as 0. Note that this strategy is optimal and independent of M, as long as the samples are classified in random order.

As Equation (5.7) shows, the worst-case scenario for the Accuracy is maximised for $\theta^* = \frac{1}{2}$ if M is even or $\theta^* \in \left\{\frac{M-1}{2M}, \frac{M+1}{2M}\right\}$ if M is odd. The corresponding Dutch Draw classifier can be transformed to a classifier that alternately predicts 0 and 1. Equivalently, $\theta^* \in \{0, 1\}$ minimises the worst-case scenario. Thus, always predicting positive or negative will give the minimal worst-case expected accuracy, namely zero.

Summary

In the previous three sections, the three branches of the Dutch Draw framework were examined (see Figure 5.1). In most cases, the optimal Dutch Draw classifier is shown to be independent of either P or M. Table 5.4 thus always holds for most measures. Only for Acc_θ, FM_θ, and $\mathrm{F}_\theta^{(\beta)}$ the results change depending on the knowledge of P and M. The summarised results are presented in Figure 5.2.

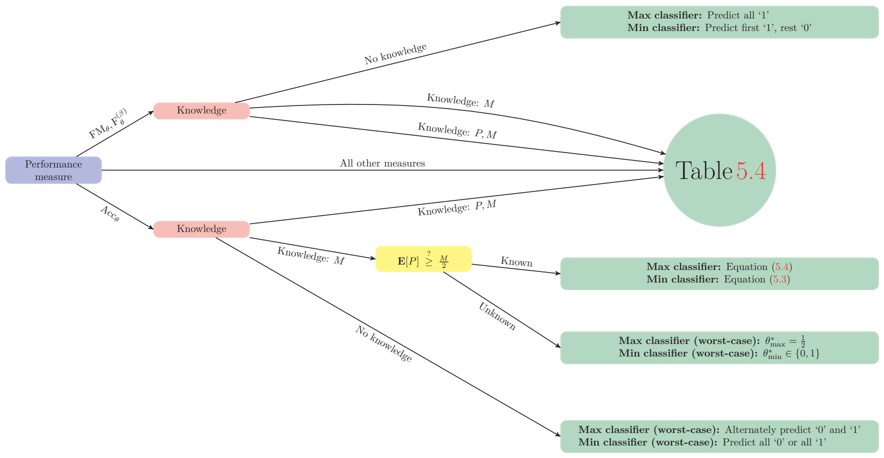

Performance measure
$FM_\theta, F_\theta^{(\beta)}$
Acc_θ
Knowledge
Knowledge
No knowledge
Knowledge: M
Knowledge: P, M
All other measures
Knowledge: P, M
Knowledge: M
No knowledge
$\mathbf{E}[P] \overset{?}{\geq} \frac{M}{2}$
Known
Unknown
Table 5.4
Max classifier: Predict all '1' **Min classifier:** Predict first '1', rest '0'
Max classifier: Equation (5.4) **Min classifier:** Equation (5.3)
Max classifier (worst-case): $\theta^*_{\max} = \frac{1}{2}$ **Min classifier (worst-case):** $\theta^*_{\min} \in \{0, 1\}$
Max classifier (worst-case): Alternately predict '0' and '1' **Min classifier (worst-case):** Predict all '0' or all '1'

Figure 5.2: Summary of Section 5.3.2: The optimal Dutch Draw classifier is given for each metric and provided assumed knowledge

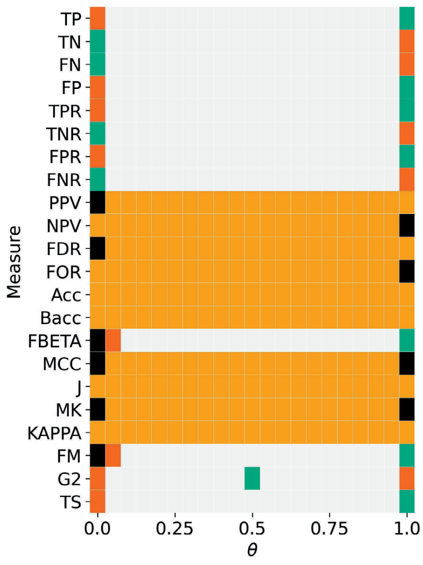
5.4 Experimental Results

In this section, we provide graphical support for the use of the Dutch Draw methodology and we show the importance of using the baseline in practice. Firstly, diagrams with the optimal values for θ for all considered evaluation metrics are presented. Secondly, for a set of commonly used datasets, the Dutch Draw baselines are calculated to show what the lower bounds are for the considered performances in practice. Lastly, we applied the Dutch Draw to many binary classifiers from literature to demonstrate that in several settings the ML methods did not outperform our baseline.

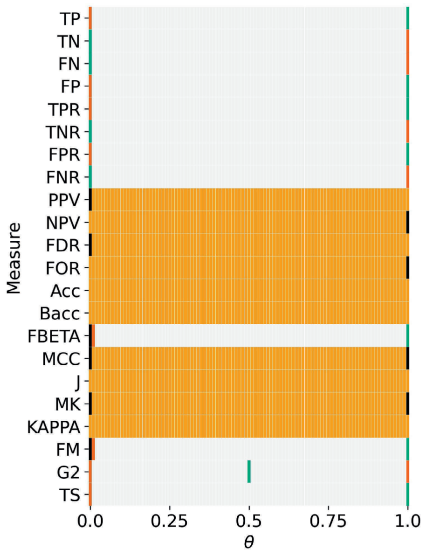
5.4.1 Visual Comparison of Evaluation Metrics

As the baseline depends on the values of P and N, different values result in different baselines. Figure 5.3 presents a visualisation of Table 5.3 for four settings of the test data class distribution and size. We consider scenario **(a)** to explain what the four diagrams convey. This scenario consists of a few observations ($M = 20$) in perfect balance ($P = N$). The horizontal axis shows the possible values of $\theta \in [0, 1]$. The vertical axis shows all the evaluation metrics and is therefore purely categorical: there is no nominal ordering between the measures. For every evaluation metric, the status for each value of θ is shown: the value is not allowed (black), the value leads to the minimal expectation (red), the value generates both the minimal and maximal expectation (orange) (i.e., the expectation is constant in θ), the value leads to neither the minimum nor the maximum (grey), or the value generates the maximal expectation (green). The diagrams show that the horizontal axis is partitioned in several rectangles. For scenario (**a**), there are 21 rectangles per measure. These correspond to the possible values of θ^*, the discretised version of θ. Remember that θ^* takes values in the set $\Theta^* = \{0, \frac{1}{M}, \ldots, \frac{M-1}{M}, 1\}$. Thus, there are $|\Theta^*| = M + 1$ unique values, and hence, 21 rectangles in this scenario.

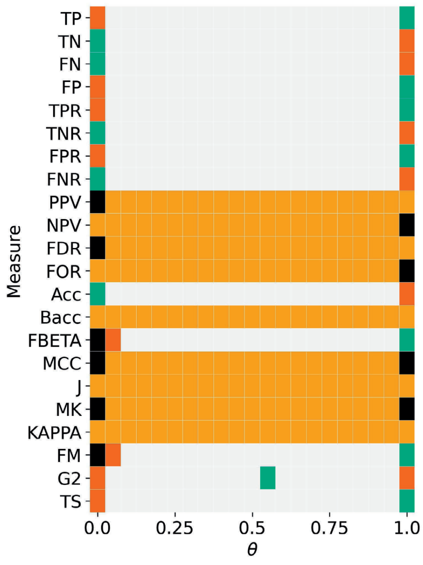
Now, in Figure 5.3, we can clearly see for each evaluation metric which values of θ lead to a minimum or maximum. In most cases, the optimal values are attained for either the extreme values of θ or all allowed values (meaning that the expectation is constant in θ). One exception to this is $\mathrm{G}_\theta^{(2)}$ (G2), as this measure is maximised for $\theta^* = \frac{1}{2}$. Note that we are not able to explicitly calculate the maximiser for $\mathrm{G}_\theta^{(2)}$, as the '✗' in Table 5.4 shows. Scenario **(b)**, in which M is chosen larger, shows the same patterns as **(a)** with more fine-grained rectangles. For scenarios **(c)** and **(d)**, the dataset is made unbalanced by choosing $P < N$. This changes the results for the Accuracy (ACC), since it is not optimised any more for every θ, but everything should be predicted according to the majority class, which is the negative class in these scenarios. Also, the maximiser for $\mathrm{G}_\theta^{(2)}$ is around 0.5 for all four settings, suggesting that $\theta^*_{\max} \approx 0.5$ in general.



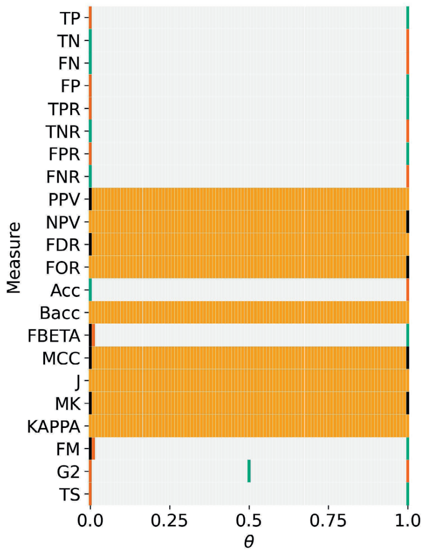
(a) Balanced classes, small $M = 20$



(b) Balanced classes, large $M = 100$



(c) Unbalanced classes, small $M = 20$



(d) Unbalanced classes, large $M = 100$

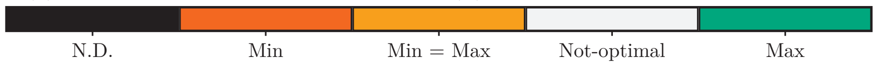


Figure 5.3: ***Visualisation of optimiser values:*** Different scenarios are considered. In the balanced setting, the ratio P:N is 1:1. In the unbalanced case, it is 1:3

5.4.2 Baselines for Existing Binary Classification Datasets

Table 5.5: *Dutch Draw baseline for UCI datasets*: Each dataset has different P and M, resulting in different Dutch Draw baselines

Measure	Adult	Bank Marketing	Banknote Authentication	Cleveland Heart Disease	Haberman's Survival	LSVT Voice Rehabilitation	Occupancy Detection	Wisconsin Cancer (Diagnostic)
TP	11687	5289	610	139	81	42	4750	212
FP	11687	5289	610	139	81	42	4750	212
FN	37155	39922	672	164	225	84	15810	357
TN	37155	39922	762	164	225	84	15810	357
TPR	1	1	1	1	1	1	1	1
FPR	1	1	1	1	1	1	1	1
FNR	1	1	1	1	1	1	1	1
TNR	1	1	1	1	1	1	1	1
PPV	0.239	0.117	0.445	0.459	0.265	0.333	0.231	0.373
NPV	0.761	0.883	0.555	0.541	0.735	0.667	0.769	0.627
FDR	0.761	0.883	0.555	0.541	0.735	0.667	0.769	0.627
FOR	0.239	0.117	0.445	0.459	0.265	0.333	0.231	0.373
F_1	0.386	0.209	0.616	0.629	0.419	0.5	0.375	0.543
J	0	0	0	0	0	0	0	0
MK	0	0	0	0	0	0	0	0
Acc	0.761	0.883	0.555	0.541	0.735	0.667	0.769	0.627
BAcc	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
MCC	0	0	0	0	0	0	0	0
κ	0	0	0	0	0	0	0	0
FM	0.489	0.342	0.667	0.677	0.514	0.577	0.481	0.61
$G^{(2)}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
TS	0.239	0.117	0.445	0.459	0.265	0.333	0.231	0.373

To show what the Dutch Draw baseline is in practice, it was calculated for all evaluation metrics on eight datasets used for binary classification. The selected datasets were extracted from the UCI Machine Learning archive [24]. The datasets are *Adult* [24], *Bank Marketing* [76], *Banknote Authentication* [24], *Cleveland Heart Disease* constructed by R. Detrano [24], *Haberman's Survival* [24], *LSVT Voice Rehabilitation* [117], *Occupancy Detection* [10], and *Wisconsin Cancer* [24]. For each dataset, the Dutch Draw baseline was determined using only P and M. The baseline can then be used to evaluate the performance of a new model. We observe in Table 5.5 that some combinations of performance measure and dataset already achieve high scores. Of course, the baseline for metrics such as TPR, TNR, FNR, and FPR always obtain a perfect score. This is because they are not *complete* binary performance measures. A complete measure is a performance measure that values both correctly predicting positives and correctly predicting negatives. For example, TPR only concerns itself with correctly predicting positive observations. Hence, predicting everything as positive irrespective of the number of negative observations always results in a perfect score of 1. Therefore, consider for example the complete measure F_1 (F_β with $\beta = 1$) on the dataset *Bank Marketing*. It already achieves a performance of 0.629, which means that any new prediction model scoring lower than this score should be reconsidered.

5.4.3 Dutch Draw Baseline Versus Existing Binary Classifiers

In the previous subsection, we presented the explicit Dutch Draw baselines on eight different datasets. These datasets had different values of P and M leading to different values for the baselines. We observed that some complete performance measures already achieved intuitively decent scores. Therefore, in this subsection, we compare these Dutch Draw baselines to the performance of many binary prediction models. Zhou and Lai [137] provide a good overview of the performance of multiple classifiers on fourteen binary datasets. The measure that the authors consider are the True Positive Rate, True Negative Rate, and Accuracy. We used these three measures to determine the four base measures, and P and N, such that we were able to calculate all evaluation metrics and the corresponding baselines. As a side note, the scores that the authors present were obtained by 10-fold cross-validation, so the base measures that we obtained are not necessarily integer. Figure 5.4 gives a visual overview of the comparisons of the obtained performance scores and the corresponding baselines for the Balanced Accuracy and F_1 score. These performance measures were chosen because they are complete and to limit the number of figures. Both axes are categorical; there is no nominal relation between the values. The horizontal axis shows the fourteen datasets and the vertical axis the prediction models that Zhou and Lai considered. A green combination of dataset and classifier means that the performance was better than the Dutch Draw baseline. A yellow combination means that the classifier performed on par with the baseline. A red combination means that the model performed worse than the baseline. Lastly, a black combination means the prediction model was not applied to the corresponding dataset. The percentages above and beside the two plots indicate the fraction of green combinations compared to the total number of green, yellow and red combinations in the corresponding column or row. Since all considered binary prediction models are supposed to learn from the data, we presumed that there would be no yellow and red combinations. After all, the Dutch Draw does not learn anything from the features, and should therefore perform worse. It is fascinating to see that there are many yellow and red combinations, especially for F_1. This raises the question of what these binary classifiers actually learned from the data. Moreover, the figure clearly shows the importance of the Dutch Draw baseline for the evaluation of binary classification problems.

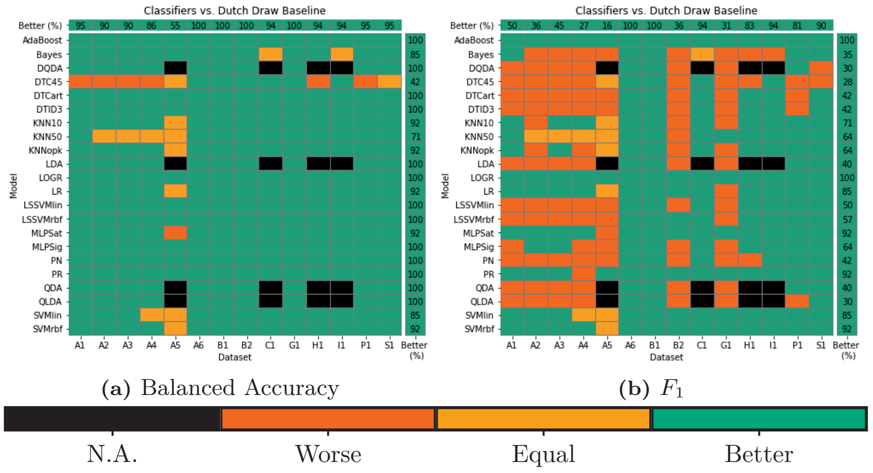


Figure 5.4: Comparison of performances of prediction models on fourteen datasets with Dutch Draw: There are four situations that can occur: the classification model is better than (green), equal to (yellow), or worse than (red) the corresponding baseline; or the combination of classifier and dataset is not available (black)

5.5 Discussion

5.5.1 Conclusion

In this research, we proposed the Dutch Draw methodology. We explained why this approach leads to a universal baseline for a given evaluation metric in any binary classification setting. A baseline is used to provide a threshold for the performance of a newly developed prediction model. However, before our research, it was often not clear what baseline should be selected. Usually, a state-of-the-art model was chosen, but what method this was could quickly change due to progress made in the field. Choosing another model could yield a very different baseline. However, choosing an elementary model such as a dummy classifier is so simple that its performance does not yield insightful information. Therefore, we proposed a method that provides a baseline that is general, simple, and informative. Hence, this Dutch Draw baseline can be seen as a strong addition that allows for comparing results across different research domains and papers.

The Dutch Draw procedure is in essence quite simple. In a set of M observation with P positives, a random sample of size $\hat{P}$ is drawn. The observations within this sample are predicted as positive and the remaining observations are predicted as negative. Since predicting is performed on the evaluation set at once, there is a strong dependence between the four base measures. This means that each base measure can be written as a linear function of TP. This is a crucial property of the Dutch Draw. Consequently, almost all the performance measures that we consider can also be written in linear terms of TP. This makes it possible to derive the baseline as the optimal expected value of the measure under the Dutch Draw approach. Our baseline is *(i)* usable in any binary classification problem, *(ii)* non-trainable and parameter-free, *(iii)* more informative than any dummy classifier, and *(iv)* an explainable minimal evaluation requirement for any new model.

In this research, we explicitly calculated the universal Dutch Draw baseline for an extensive list of evaluation metrics. We were not able to do this for the G-mean 2 and the Threat Score, but they can still be determined with simulation. This greatly extends the applicability of our research. Furthermore, we expanded the analysis of the Dutch Draw by considering additional mathematical properties. Firstly, we provided the distribution and optimisation values of each evaluation metric. Secondly, we considered the consequences of the case in which the number of positives P was unknown and even the case in which the number of observations M was unknown. Also, we applied our Dutch Draw method to several well-known binary datasets to obtain the universal baselines, and we compared these baselines to the scores that several ML models in various studies obtained. We observed that for multiple performance measures, the ML methods did not outperform the baseline on several datasets. Hence, such models should be reconsidered or not applied to these datasets. This makes the Dutch Draw baseline a relevant tool in the development of new ML methods.

In summary, we have provided the reader a standardised baseline that has many desirable properties for many evaluation metrics in binary classification. For quick access, see Table 5.3 for properties of the Dutch Draw, and see Table 5.4 for the baselines and corresponding optimisation values.

5.5.2 Further Research

Our baseline is a stepping stone for further research, where multiple avenues should be explored. We discuss four possible research directions.

Firstly, we are now able to determine whether a binary classification model performs better than a standardised baseline. However, we do not yet know how *much* it performs better (or worse). For example, let the baseline have a score of 0.5 and a new model a score of 0.9. How much better is the latter score? It could be that a tiny bit of extra information already pushes the score from 0.5 to 0.9. Or, it is possible that a model needs a lot of information to understand the intricacies of the problem, making it very hard to reach a score of 0.9. Thus, it is necessary to quantify how hard it is to reach any score. Also, when another model is added that achieves a score of 0.91, can the difference in performance of these models be quantified? Is it only a slightly better model or is it a leap forward?

Secondly, our Dutch Draw baseline could be used to construct new standardised evaluation metrics from their original versions. The advantage of these new measures would be that the interpretation of their scores is independent of the number of positive and negative observations in the dataset. In other words, the Dutch Draw baseline would already be incorporated in the new measure, such that comparing a score to the baseline is not necessary any more.

Thirdly, another natural extension would be to drop the binary assumption and consider multi-class classification. This is more complicated than it seems, because not every multi-class evaluation metric follows automatically from its binary counterpart. However, we expect that for most multi-class measures it is again optimal to always predict a single specific class.

Fourthly, the spirit of the Dutch Draw could be used to create universal baselines for other prediction problems, such as regression. This means a standardised approach that also uses (almost) no information from the data and is able to generate a measure-specific baseline to which newly developed models could be compared.

5.6 Appendix

This section contains the complete theoretical analysis that is used to gather the information presented in Sections 5.2 and 5.3, and more specifically, Tables 5.2, 5.3 and 5.4. Each subsection is dedicated to one of the evaluation metrics. The following definitions are frequently used throughout this section:

$$\begin{aligned} Z_\theta(a,b) &:= a \cdot \mathrm{TP}_\theta + b \text{ with } a, b \in \mathbb{R} \\ H_\theta(a,b) &:= \text{probability distribution of } Z_\theta(a,b). \end{aligned}$$

5.6.1 Number of True Positives

The *Number of True Positives* TP_θ is one of the four base measures that are introduced in Section 5.2.2. This measure indicates how many of the predicted positive observations are actually positive. Under the Dutch Draw methodology, each evaluation metric can be written in terms of TP_θ.

Definition and Distribution

Since we want to formulate each measure in terms of TP_θ, we have for TP_θ:

$$\mathrm{TP}_\theta \overset{(B1)}{=} Z_\theta(1,0) \sim H_\theta(1,0).$$

The range of this base measure depends on θ. Therefore, Equation (R) yields the range of this measure:

$$\mathrm{TP}_\theta \in \mathcal{R}(Z_\theta(1,0)).$$

Expectation

The expectation of TP_θ using the Dutch Draw is given by

$$\mathbf{E}[\text{TP}_\theta] = \mathbf{E}[Z_\theta\,(1,0)] \overset{(E)}{=} \frac{\lfloor M \cdot \theta \rceil}{M} \cdot P = \theta^* \cdot P. \tag{5.9}$$

Optimal Baselines

The Dutch Draw baseline is given by the optimal expectation. Equation (5.9) shows that the expected value depends on the parameter θ. Therefore, either the minimum or maximum of the expectation yields the baseline. They are given by

$$\min_{\theta\in[0,1]} (\mathbf{E}[\text{TP}_\theta]) = P \cdot \min_{\theta\in[0,1]} \left(\frac{\lfloor M \cdot \theta \rceil}{M}\right) = 0,$$
$$\max_{\theta\in[0,1]} (\mathbf{E}[\text{TP}_\theta]) = P \cdot \max_{\theta\in[0,1]} \left(\frac{\lfloor M \cdot \theta \rceil}{M}\right) = P.$$

The values of $\theta \in [0,1]$ that minimise or maximise the expected value are θ_{min} and θ_{max}, respectively, and are defined as

$$\theta_{\text{min}} \in \operatorname*{arg\,min}_{\theta\in[0,1]} (\mathbf{E}[\text{TP}_\theta]) = \operatorname*{arg\,min}_{\theta\in[0,1]} \left(\frac{\lfloor M \cdot \theta \rceil}{M}\right) = \left[0, \frac{1}{2M}\right),$$
$$\theta_{\text{max}} \in \operatorname*{arg\,max}_{\theta\in[0,1]} (\mathbf{E}[\text{TP}_\theta]) = \operatorname*{arg\,max}_{\theta\in[0,1]} \left(\frac{\lfloor M \cdot \theta \rceil}{M}\right) = \left[1 - \frac{1}{2M}, 1\right].$$

Equivalently, the discrete optimisers $\theta^*_{\text{min}} \in \Theta^*$ and $\theta^*_{\text{max}} \in \Theta^*$ are determined by

$$\theta^*_{\text{min}} \in \operatorname*{arg\,min}_{\theta^*\in\Theta^*} \{\mathbf{E}[\text{TP}_{\theta^*}]\} = \operatorname*{arg\,min}_{\theta^*\in\Theta^*} \{\theta^*\} = \{0\},$$
$$\theta^*_{\text{max}} \in \operatorname*{arg\,max}_{\theta^*\in\Theta^*} \{\mathbf{E}[\text{TP}_{\theta^*}]\} = \operatorname*{arg\,max}_{\theta^*\in\Theta^*} \{\theta^*\} = \{1\}.$$

5.6.2 Number of False Positives

The *Number of False Positives* FP_θ is one of the four base measures that we discussed in Section 5.2.2. This base measure counts the number of mistakes made by predicting instances positive while the actual labels are negative.

Definition and Distribution

Each base measure can be expressed in terms of TP_θ, thus we have for FP_θ:

$$\text{FP}_\theta \overset{(B2)}{=} \lfloor M \cdot \theta \rceil - \text{TP}_\theta = Z_\theta\,(-1, \lfloor M \cdot \theta \rceil) \sim H_\theta\,(-1, \lfloor M \cdot \theta \rceil)\,,$$

and for its range:

$$\text{FP}_\theta \overset{(R)}{\in} \mathcal{R}\,(Z_\theta\,(-1, \lfloor M \cdot \theta \rceil))\,.$$

Expectation

As Equation (B2) shows, FP_θ is linear in TP_θ with slope $a = -1$ and intercept $b = \lfloor M \cdot \theta \rceil$, thus the expectation of FP_θ is defined as

$$\begin{aligned}\mathbf{E}[\text{FP}_\theta] &= \mathbf{E}[Z_\theta(-1, \lfloor M \cdot \theta \rceil)] \overset{(\text{E})}{=} -1 \cdot \mathbf{E}[\text{TP}_\theta] + \lfloor M \cdot \theta \rceil = \frac{\lfloor M \cdot \theta \rceil}{M} \cdot (M - P) \\ &= \theta^* \cdot (M - P).\end{aligned}$$

Optimal Baselines

The baselines of FP_θ are given by the extreme values of its expectation. Hence:

$$\begin{aligned}\min_{\theta \in [0,1]} (\mathbf{E}[\text{FP}_\theta]) &= (M - P) \min_{\theta \in [0,1]} \left(\frac{\lfloor M \cdot \theta \rceil}{M} \right) = 0, \\ \max_{\theta \in [0,1]} (\mathbf{E}[\text{FP}_\theta]) &= (M - P) \max_{\theta \in [0,1]} \left(\frac{\lfloor M \cdot \theta \rceil}{M} \right) = M - P.\end{aligned}$$

The corresponding optimisation values $\theta_{\text{min}} \in [0, 1]$ and $\theta_{\text{max}} \in [0, 1]$ are

$$\begin{aligned}\theta_{\text{min}} &\in \arg\min_{\theta \in [0,1]} (\mathbf{E}[\text{FP}_\theta]) = \arg\min_{\theta \in [0,1]} \left(\frac{\lfloor M \cdot \theta \rceil}{M} \right) = \left[0, \frac{1}{2M}\right), \\ \theta_{\text{max}} &\in \arg\max_{\theta \in [0,1]} (\mathbf{E}[\text{FP}_\theta]) = \arg\max_{\theta \in [0,1]} \left(\frac{\lfloor M \cdot \theta \rceil}{M} \right) = \left[1 - \frac{1}{2M}, 1\right].\end{aligned}$$

The discrete versions $\theta^*_{\text{min}} \in \Theta^*$ and $\theta^*_{\text{max}} \in \Theta^*$ of the optimisation values are determined by

$$\begin{aligned}\theta^*_{\text{min}} &\in \arg\min_{\theta^* \in \Theta^*} \{\mathbf{E}[\text{FP}_{\theta^*}]\} = \arg\min_{\theta^* \in \Theta^*} \{\theta^*\} = \{0\}, \\ \theta^*_{\text{max}} &\in \arg\max_{\theta^* \in \Theta^*} \{\mathbf{E}[\text{FP}_{\theta^*}]\} = \arg\max_{\theta^* \in \Theta^*} \{\theta^*\} = \{1\}.\end{aligned}$$

5.6.3 Number of False Negatives

The *Number of False Negative* FN_θ is one of the four base measures that are introduced in Section 5.2.2. This base measure counts the number of mistakes made by predicting instances negative while the actual labels are positive.

Definition and Distribution

Equation (B3) shows that FN_θ can be expressed in terms of TP_θ:

$$\text{FN}_\theta \overset{(\text{B3})}{=} P - \text{TP}_\theta = Z_\theta(-1, P) \sim H_\theta(-1, P),$$

and for its range:

$$\text{FN}_\theta \overset{(\text{R})}{\in} \mathcal{R}(Z_\theta(-1, P)).$$

Expectation

As Equation (B3) shows, FN_θ is linear in TP_θ with slope $a = -1$ and intercept $b = P$. Hence, the expectation of FN_θ is given by

$$\begin{aligned}\mathbf{E}[\mathrm{FN}_\theta] = \mathbf{E}[Z_\theta(-1, P)] &\overset{(\mathrm{E})}{=} -1 \cdot \mathbf{E}[\mathrm{TP}_\theta] + P = \left(1 - \frac{\lfloor M \cdot \theta \rceil}{M}\right) \cdot P \\ &= (1 - \theta^*) \cdot P.\end{aligned}$$

Optimal Baselines

The range of the expectation of FN_θ determines the baselines. The extreme values are given by

$$\begin{aligned}\min_{\theta \in [0,1]} (\mathbf{E}[\mathrm{FN}_\theta]) &= P \cdot \min_{\theta \in [0,1]} \left(1 - \frac{\lfloor M \cdot \theta \rceil}{M}\right) = 0, \\ \max_{\theta \in [0,1]} (\mathbf{E}[\mathrm{FN}_\theta]) &= P \cdot \max_{\theta \in [0,1]} \left(1 - \frac{\lfloor M \cdot \theta \rceil}{M}\right) = P.\end{aligned}$$

The associated optimisation values $\theta_{\min} \in [0, 1]$ and $\theta_{\max} \in [0, 1]$ are then

$$\begin{aligned}\theta_{\min} &\in \operatorname*{arg\,min}_{\theta \in [0,1]} (\mathbf{E}[\mathrm{FN}_\theta]) = \operatorname*{arg\,min}_{\theta \in [0,1]} \left(1 - \frac{\lfloor M \cdot \theta \rceil}{M}\right) = \left[1 - \frac{1}{2M}, 1\right], \\ \theta_{\max} &\in \operatorname*{arg\,max}_{\theta \in [0,1]} (\mathbf{E}[\mathrm{FN}_\theta]) = \operatorname*{arg\,max}_{\theta \in [0,1]} \left(1 - \frac{\lfloor M \cdot \theta \rceil}{M}\right) = \left[0, \frac{1}{2M}\right),\end{aligned}$$

respectively. The discrete versions $\theta^*_{\min} \in \Theta^*$ and $\theta^*_{\max} \in \Theta^*$ of the optimisers are as follows:

$$\begin{aligned}\theta^*_{\min} &\in \operatorname*{arg\,min}_{\theta^* \in \Theta^*} \{\mathbf{E}[\mathrm{FN}_{\theta^*}]\} = \operatorname*{arg\,min}_{\theta^* \in \Theta^*} \{1 - \theta^*\} = \{1\}, \\ \theta^*_{\max} &\in \operatorname*{arg\,max}_{\theta^* \in \Theta^*} \{\mathbf{E}[\mathrm{FN}_{\theta^*}]\} = \operatorname*{arg\,max}_{\theta^* \in \Theta^*} \{1 - \theta^*\} = \{0\}.\end{aligned}$$

5.6.4 Number of True Negatives

The *Number of True Negatives* TN_θ is also one of the four base measures and is introduced in Section 5.2.2. This base measure counts the number of negative predicted instances that are actually negative.

Definition and Distribution

Since we want to formulate each measure in terms of TP_θ, we have for TN_θ:

$$\mathrm{TN}_\theta = M - P - \lfloor M \cdot \theta \rceil + \mathrm{TP}_\theta,$$

which corresponds to Equation (B4). Furthermore,

$$\mathrm{TN}_\theta \overset{(\mathrm{B4})}{=} Z_\theta(1, M - P - \lfloor M \cdot \theta \rceil) \sim H_\theta(1, M - P - \lfloor M \cdot \theta \rceil),$$

and for its range

$$\mathrm{TN}_\theta \overset{(\mathrm{R})}{\in} \mathcal{R}\left(Z_\theta\left(1, M-P-\lfloor M\cdot\theta\rceil\right)\right).$$

Expectation

TN_θ is linear in TP_θ with slope $a=1$ and intercept $b = M-P-\lfloor M\cdot\theta\rceil$, so its expectation is given by

$$\begin{aligned}\mathbf{E}[\mathrm{TN}_\theta] = \mathbf{E}[Z_\theta\left(1, M-P-\lfloor M\cdot\theta\rceil\right)] &\overset{(\mathrm{E})}{=} 1\cdot\mathbf{E}[\mathrm{TP}_\theta] + M-P-\lfloor M\cdot\theta\rceil,\\ &= \left(1-\frac{\lfloor M\cdot\theta\rceil}{M}\right)(M-P) = (1-\theta^*)\,(M-P).\end{aligned}$$

Optimal Baselines

To determine the range of the expectation of TN_θ, and hence, obtain baselines, its extreme values are calculated:

$$\begin{aligned}\min_{\theta\in[0,1]}\left(\mathbf{E}[\mathrm{TN}_\theta]\right) &= (M-P)\min_{\theta\in[0,1]}\left(1-\frac{\lfloor M\cdot\theta\rceil}{M}\right) = 0,\\ \max_{\theta\in[0,1]}\left(\mathbf{E}[\mathrm{TN}_\theta]\right) &= (M-P)\max_{\theta\in[0,1]}\left(1-\frac{\lfloor M\cdot\theta\rceil}{M}\right) = M-P.\end{aligned}$$

The associated optimisation values $\theta_{\mathrm{min}}\in[0,1]$ and $\theta_{\mathrm{max}}\in[0,1]$ are

$$\begin{aligned}\theta_{\mathrm{min}} &\in \operatorname*{arg\,min}_{\theta\in[0,1]}\left(\mathbf{E}[\mathrm{TN}_\theta]\right) = \operatorname*{arg\,min}_{\theta\in[0,1]}\left(1-\frac{\lfloor M\cdot\theta\rceil}{M}\right) = \left[1-\frac{1}{2M}, 1\right],\\ \theta_{\mathrm{max}} &\in \operatorname*{arg\,max}_{\theta\in[0,1]}\left(\mathbf{E}[\mathrm{TN}_\theta]\right) = \operatorname*{arg\,max}_{\theta\in[0,1]}\left(1-\frac{\lfloor M\cdot\theta\rceil}{M}\right) = \left[0, \frac{1}{2M}\right).\end{aligned}$$

The discrete equivalents $\theta^*_{\mathrm{min}}\in\Theta^*$ and $\theta^*_{\mathrm{max}}\in\Theta^*$ are then determined by

$$\begin{aligned}\theta^*_{\mathrm{min}} &\in \operatorname*{arg\,min}_{\theta^*\in\Theta^*}\left\{\mathbf{E}[\mathrm{TN}_{\theta^*}]\right\} = \operatorname*{arg\,min}_{\theta^*\in\Theta^*}\left\{1-\theta^*\right\} = \{1\},\\ \theta^*_{\mathrm{max}} &\in \operatorname*{arg\,max}_{\theta^*\in\Theta^*}\left\{\mathbf{E}[\mathrm{TN}_{\theta^*}]\right\} = \operatorname*{arg\,max}_{\theta^*\in\Theta^*}\left\{1-\theta^*\right\} = \{0\}.\end{aligned}$$

5.6.5 True Positive Rate

The *True Positive Rate* TPR_θ, *Recall*, or *Sensitivity* is the performance measure that presents the fraction of positive observations that are correctly predicted. This makes it a fundamental performance measure in binary classification.

Definition and Distribution

The True Positive Rate is commonly defined as

$$\mathrm{TPR}_\theta = \frac{\mathrm{TP}_\theta}{P}. \tag{5.10}$$

Hence, $P > 0$ should hold, otherwise the denominator is zero. Now, TPR_θ is linear in TP_θ and can therefore be written as

$$\mathrm{TPR}_\theta = Z_\theta\left(\frac{1}{P}, 0\right) \sim H_\theta\left(\frac{1}{P}, 0\right), \tag{5.11}$$

and for its range:

$$\mathrm{TPR}_\theta \overset{(\mathrm{R})}{\in} \mathcal{R}\left(Z_\theta\left(\frac{1}{P}, 0\right)\right).$$

Expectation

Since TPR_θ is linear in TP_θ with slope $a = 1/P$ and intercept $b = 0$, its expectation is

$$\mathbf{E}[\mathrm{TPR}_\theta] = \mathbf{E}\left[Z_\theta\left(\frac{1}{P}, 0\right)\right] \overset{(\mathrm{E})}{=} \frac{1}{P} \cdot \mathbf{E}[\mathrm{TP}_\theta] + 0 = \frac{\lfloor M \cdot \theta \rceil}{M} = \theta^*.$$

Optimal Baselines

The range of the expectation of TPR_θ directly determines the baselines. The extreme values are given by

$$\min_{\theta \in [0,1]} \left(\mathbf{E}[\mathrm{TPR}_\theta]\right) = \min_{\theta \in [0,1]} \left(\frac{\lfloor M \cdot \theta \rceil}{M}\right) = 0,$$
$$\max_{\theta \in [0,1]} \left(\mathbf{E}[\mathrm{TPR}_\theta]\right) = \max_{\theta \in [0,1]} \left(\frac{\lfloor M \cdot \theta \rceil}{M}\right) = 1.$$

Furthermore, the corresponding optimisation values $\theta_{\min} \in [0, 1]$ and $\theta_{\max} \in [0, 1]$ are given by

$$\theta_{\min} \in \underset{\theta \in [0,1]}{\arg\min} \left(\mathbf{E}[\mathrm{TPR}_\theta]\right) = \underset{\theta \in [0,1]}{\arg\min} \left(\frac{\lfloor M \cdot \theta \rceil}{M}\right) = \left[0, \frac{1}{2M}\right),$$
$$\theta_{\max} \in \underset{\theta \in [0,1]}{\arg\max} \left(\mathbf{E}[\mathrm{TPR}_\theta]\right) = \underset{\theta \in [0,1]}{\arg\max} \left(\frac{\lfloor M \cdot \theta \rceil}{M}\right) = \left[1 - \frac{1}{2M}, 1\right].$$

The discrete versions $\theta^*_{\min} \in \Theta^*$ and $\theta^*_{\max} \in \Theta^*$ of the optimisers are then

$$\theta^*_{\min} \in \underset{\theta^* \in \Theta^*}{\arg\min} \left\{\mathbf{E}[\mathrm{TPR}_{\theta^*}]\right\} = \underset{\theta^* \in \Theta^*}{\arg\min} \left\{\theta^*\right\} = \{0\},$$
$$\theta^*_{\max} \in \underset{\theta^* \in \Theta^*}{\arg\max} \left\{\mathbf{E}[\mathrm{TPR}_{\theta^*}]\right\} = \underset{\theta^* \in \Theta^*}{\arg\max} \left\{\theta^*\right\} = \{1\},$$

respectively.

5.6.6 False Positive Rate

The *False Positive Rate* FPR_θ or *Fall-out* is the performance measure that shows the fraction of incorrectly predicted negative observations. Hence, it can be seen as the counterpart to the True Negative Rate that is introduced in Section 5.6.8.

Definition and Distribution

The False Positive Rate is commonly defined as

$$\text{FPR}_\theta = \frac{\text{FP}_\theta}{N}.$$

Hence, $N := M - P$ should hold, otherwise the denominator is zero. By using Equation (B2), FPR_θ can be restated as

$$\text{FPR}_\theta = \frac{\lfloor M \cdot \theta \rceil - \text{TP}_\theta}{M - P}. \tag{5.12}$$

Note that it is linear in TP_θ and can therefore be written as

$$\text{FPR}_\theta = Z_\theta\left(-\frac{1}{M-P}, \frac{\lfloor M \cdot \theta \rceil}{M-P}\right) \sim H_\theta\left(-\frac{1}{M-P}, \frac{\lfloor M \cdot \theta \rceil}{M-P}\right),$$

with range:

$$\text{FPR}_\theta \overset{(\text{R})}{\in} \mathcal{R}\left(Z_\theta\left(-\frac{1}{M-P}, \frac{\lfloor M \cdot \theta \rceil}{M-P}\right)\right).$$

Expectation

Since FPR_θ is linear in TP_θ with slope $a = -1/(M-P)$ and intercept $b = \lfloor M \cdot \theta \rceil / (M-P)$, its expectation is given by

$$\begin{aligned}\mathbf{E}[\text{FPR}_\theta] = \mathbf{E}\left[Z_\theta\left(-\frac{1}{M-P}, \frac{\lfloor M \cdot \theta \rceil}{M-P}\right)\right] &\overset{(\text{E})}{=} -\frac{1}{M-P} \cdot \mathbf{E}[\text{TP}_\theta] + \frac{\lfloor M \cdot \theta \rceil}{M-P} \\ &= \frac{\lfloor M \cdot \theta \rceil}{M} = \theta^*.\end{aligned}$$

Optimal Baselines

The extreme values of the expectation of FPR_θ determine the baselines. The range is given by

$$\min_{\theta \in [0,1]} (\mathbf{E}[\text{FPR}_\theta]) = \min_{\theta \in [0,1]} \left(\frac{\lfloor M \cdot \theta \rceil}{M}\right) = 0,$$
$$\max_{\theta \in [0,1]} (\mathbf{E}[\text{FPR}_\theta]) = \max_{\theta \in [0,1]} \left(\frac{\lfloor M \cdot \theta \rceil}{M}\right) = 1.$$

Moreover, the optimisers $\theta_{\min} \in [0,1]$ and $\theta_{\max} \in [0,1]$ for the extreme values are determined by

$$\theta_{\min} \in \underset{\theta\in[0,1]}{\arg\min}\left(\mathbf{E}[\mathrm{FPR}_\theta]\right) = \underset{\theta\in[0,1]}{\arg\min}\left(\frac{\lfloor M\cdot\theta\rceil}{M}\right) = \left[0, \frac{1}{2M}\right),$$
$$\theta_{\max} \in \underset{\theta\in[0,1]}{\arg\max}\left(\mathbf{E}[\mathrm{FPR}_\theta]\right) = \underset{\theta\in[0,1]}{\arg\max}\left(\frac{\lfloor M\cdot\theta\rceil}{M}\right) = \left[1-\frac{1}{2M}, 1\right],$$

respectively. The discrete forms $\theta^*_{\min} \in \Theta^*$ and $\theta^*_{\max} \in \Theta^*$ of these are then

$$\theta^*_{\min} \in \underset{\theta^*\in\Theta^*}{\arg\min}\left\{\mathbf{E}[\mathrm{FPR}_{\theta^*}]\right\} = \underset{\theta^*\in\Theta^*}{\arg\min}\left\{\theta^*\right\} = \{0\},$$
$$\theta^*_{\max} \in \underset{\theta^*\in\Theta^*}{\arg\max}\left\{\mathbf{E}[\mathrm{FPR}_{\theta^*}]\right\} = \underset{\theta^*\in\Theta^*}{\arg\max}\left\{\theta^*\right\} = \{1\}.$$

5.6.7 False Negative Rate

The *False Negative Rate* FNR_θ or *Miss Rate* is the performance measure that indicates the relative number of incorrectly predicted positive observations. Therefore, it can be seen as the counterpart to the True Positive Rate that is discussed in Section 5.6.5.

Definition and Distribution

The False Negative Rate is commonly defined as

$$\mathrm{FNR}_\theta = \frac{\mathrm{FN}_\theta}{P}.$$

Hence, $P > 0$ should hold, otherwise the denominator is zero. With the aid of Equation (B3), FNR_θ can be reformulated to

$$\mathrm{FNR}_\theta = \frac{P - \mathrm{TP}_\theta}{P} = 1 - \frac{\mathrm{TP}_\theta}{P}.$$

Thus, it is linear in TP_θ and can therefore be written as

$$\mathrm{FNR}_\theta = Z_\theta\left(-\frac{1}{P}, 1\right) \sim H_\theta\left(-\frac{1}{P}, 1\right),$$

and for its range:

$$\mathrm{FNR}_\theta \overset{(R)}{\in} \mathcal{R}\left(Z_\theta\left(-\frac{1}{P}, 1\right)\right).$$

Expectation

Because FNR_θ is linear in TP_θ with slope $a = -1/P$ and intercept $b = 1$, its expectation is

$$\mathbf{E}[\mathrm{FNR}_\theta] = \mathbf{E}\left[Z_\theta\left(-\frac{1}{P}, 1\right)\right] \overset{(E)}{=} -\frac{1}{P}\cdot\mathbf{E}[\mathrm{TP}_\theta] + 1 = 1 - \frac{\lfloor M\cdot\theta\rceil}{M} = 1 - \theta^*.$$

Optimal Baselines

The range of the expectation of FNR_θ determines the baselines. The extreme values are given by:

$$\min_{\theta\in[0,1]} (\mathbf{E}[\text{FNR}_\theta]) = \min_{\theta\in[0,1]} \left(1 - \frac{\lfloor M\cdot\theta\rceil}{M}\right) = 0,$$
$$\max_{\theta\in[0,1]} (\mathbf{E}[\text{FNR}_\theta]) = \max_{\theta\in[0,1]} \left(1 - \frac{\lfloor M\cdot\theta\rceil}{M}\right) = 1.$$

Furthermore, the optimisers $\theta_{\min} \in [0,1]$ and $\theta_{\max} \in [0,1]$ for the extreme values are as follows:

$$\theta_{\min} \in \arg\min_{\theta\in[0,1]} (\mathbf{E}[\text{FNR}_\theta]) = \arg\min_{\theta\in[0,1]} \left(1 - \frac{\lfloor M\cdot\theta\rceil}{M}\right) = \left[1-\frac{1}{2M}, 1\right],$$
$$\theta_{\max} \in \arg\max_{\theta\in[0,1]} (\mathbf{E}[\text{FNR}_\theta]) = \arg\max_{\theta\in[0,1]} \left(1 - \frac{\lfloor M\cdot\theta\rceil}{M}\right) = \left[0, \frac{1}{2M}\right),$$

respectively. The discrete versions $\theta^*_{\min} \in \Theta^*$ and $\theta^*_{\max} \in \Theta^*$ of the optimisation values are then:

$$\theta^*_{\min} \in \arg\min_{\theta^*\in\Theta^*} \{\mathbf{E}[\text{FNR}_{\theta^*}]\} = \arg\min_{\theta^*\in\Theta^*} \{1-\theta^*\} = \{1\},$$
$$\theta^*_{\max} \in \arg\max_{\theta^*\in\Theta^*} \{\mathbf{E}[\text{FNR}_{\theta^*}]\} = \arg\max_{\theta^*\in\Theta^*} \{1-\theta^*\} = \{0\}.$$

5.6.8 True Negative Rate

The *True Negative Rate* TNR_θ, *Specificity*, or *Selectivity* is the measure that shows how relatively well the negative observations are correctly predicted. Hence, this performance measure is a fundamental measure in binary classification.

Definition and Distribution

The True Negative Rate is commonly defined as

$$\text{TNR}_\theta = \frac{\text{TN}_\theta}{N}.$$

Hence, $N := M - P > 0$ should hold, otherwise the denominator is zero. By using Equation (B4), TNR_θ can be rewritten as

$$\text{TNR}_\theta = \frac{M - P - \lfloor M\cdot\theta\rceil + \text{TP}_\theta}{M-P} = 1 - \frac{\lfloor M\cdot\theta\rceil - \text{TP}_\theta}{M-P}.$$

Hence, it is linear in TP_θ and can therefore be written as

$$\text{TNR}_\theta = Z_\theta\left(\frac{1}{M-P}, 1-\frac{\lfloor M\cdot\theta\rceil}{M-P}\right) \sim H_\theta\left(\frac{1}{M-P}, 1-\frac{\lfloor M\cdot\theta\rceil}{M-P}\right), \tag{5.13}$$

and for its range:

$$\text{TNR}_\theta \overset{(\text{R})}{\in} \mathcal{R}\left(Z_\theta\left(\frac{1}{M-P}, 1-\frac{\lfloor M\cdot\theta\rceil}{M-P}\right)\right).$$

Expectation

Since TNR_θ is linear in TP_θ in terms of $Z_\theta(a,b)$ with slope $a = 1/(M-P)$ and intercept $b = 1 - \lfloor M\cdot\theta\rceil/(M-P)$, its expectation is

$$\begin{aligned}\mathbf{E}[\text{TNR}_\theta] &= \mathbf{E}\left[Z_\theta\left(\frac{1}{M-P}, 1-\frac{\lfloor M\cdot\theta\rceil}{M-P}\right)\right] \overset{(\text{E})}{=} \frac{1}{M-P}\cdot\mathbf{E}[\text{TP}_\theta] + 1 - \frac{\lfloor M\cdot\theta\rceil}{M-P}\\ &= 1-\frac{\lfloor M\cdot\theta\rceil}{M} = 1-\theta^*.\end{aligned}$$

Optimal Baselines

The extreme values of the expectation of TNR_θ determine the baselines. The range is given by

$$\begin{aligned}\min_{\theta\in[0,1]}\left(\mathbf{E}[\text{TNR}_\theta]\right) &= \min_{\theta\in[0,1]}\left(1-\frac{\lfloor M\cdot\theta\rceil}{M}\right) = 0,\\ \max_{\theta\in[0,1]}\left(\mathbf{E}[\text{TNR}_\theta]\right) &= \max_{\theta\in[0,1]}\left(1-\frac{\lfloor M\cdot\theta\rceil}{M}\right) = 1.\end{aligned}$$

Moreover, the optimisation values $\theta_{\min}\in[0,1]$ and $\theta_{\max}\in[0,1]$ corresponding to the extreme values are defined as

$$\begin{aligned}\theta_{\min} &\in \arg\min_{\theta\in[0,1]}\left(\mathbf{E}[\text{TNR}_\theta]\right) = \arg\min_{\theta\in[0,1]}\left(1-\frac{\lfloor M\cdot\theta\rceil}{M}\right) = \left[1-\frac{1}{2M}, 1\right],\\ \theta_{\max} &\in \arg\max_{\theta\in[0,1]}\left(\mathbf{E}[\text{TNR}_\theta]\right) = \arg\max_{\theta\in[0,1]}\left(1-\frac{\lfloor M\cdot\theta\rceil}{M}\right) = \left[0, \frac{1}{2M}\right),\end{aligned}$$

respectively. The discrete versions $\theta^*_{\min}\in\Theta^*$ and $\theta^*_{\max}\in\Theta^*$ of the optimisers are given by

$$\begin{aligned}\theta^*_{\min} &\in \arg\min_{\theta^*\in\Theta^*}\left\{\mathbf{E}[\text{TNR}_{\theta^*}]\right\} = \arg\min_{\theta^*\in\Theta^*}\left\{1-\theta^*\right\} = \{1\},\\ \theta^*_{\max} &\in \arg\max_{\theta^*\in\Theta^*}\left\{\mathbf{E}[\text{TNR}_{\theta^*}]\right\} = \arg\max_{\theta^*\in\Theta^*}\left\{1-\theta^*\right\} = \{0\}.\end{aligned}$$

5.6.9 Positive Predictive Value

The *Positive Predictive Value* PPV_θ or *Precision* is the performance measure that considers the fraction of all positively predicted observations that are in fact positive. Therefore, it provides an indication of how cautious the model is in assigning positive predictions. A large value means the model is cautious in predicting observations as positive, while a small value means the opposite.

Definition and Distribution

The Positive Predictive Value is commonly defined as

$$\mathrm{PPV}_\theta = \frac{\mathrm{TP}_\theta}{\mathrm{TP}_\theta + \mathrm{FP}_\theta}. \tag{5.14}$$

By using Equations (B1) and (B2), this definition can be reformulated to

$$\mathrm{PPV}_\theta = \frac{\mathrm{TP}_\theta}{\lfloor M \cdot \theta \rceil}.$$

Note that this performance measure is only defined whenever $\lfloor M \cdot \theta \rceil > 0$, otherwise the denominator is zero. Therefore, we assume specifically for PPV_θ that $\theta \geq \frac{1}{2M}$. The definition of PPV_θ is linear in TP_θ and can thus be formulated as

$$\mathrm{PPV}_\theta = Z_\theta \left(\frac{1}{\lfloor M \cdot \theta \rceil}, 0 \right) \sim H_\theta \left(\frac{1}{\lfloor M \cdot \theta \rceil}, 0 \right), \tag{5.15}$$

with range:

$$\mathrm{PPV}_\theta \overset{(\mathrm{R})}{\in} \mathcal{R} \left(Z_\theta \left(\frac{1}{\lfloor M \cdot \theta \rceil}, 0 \right) \right).$$

Expectation

Because PPV_θ is linear in TP_θ with slope $a = 1/\lfloor M \cdot \theta \rceil$ and intercept $b = 0$, its expectation is

$$\mathbf{E}[\mathrm{PPV}_\theta] = \mathbf{E}\left[Z_\theta \left(\frac{1}{\lfloor M \cdot \theta \rceil}, 0 \right) \right] \overset{(\mathrm{E})}{=} \frac{1}{\lfloor M \cdot \theta \rceil} \cdot \mathbf{E}[\mathrm{TP}_\theta] + 0 = \frac{P}{M}.$$

Optimal Baselines

The baselines are determined by the extreme values of the expectation of PPV_θ:

$$\min_{\theta \in [1/(2M), 1]} \left(\mathbf{E}[\mathrm{PPV}_\theta] \right) = \frac{P}{M},$$
$$\max_{\theta \in [1/(2M), 1]} \left(\mathbf{E}[\mathrm{PPV}_\theta] \right) = \frac{P}{M},$$

because the expectation does not depend on θ. Hence, the optimisation values $\theta_{\min}$ and $\theta_{\max}$ are simply all allowed values for θ:

$$\theta_{\min} = \theta_{\max} \in \left[\frac{1}{2M}, 1 \right].$$

Consequently, the discrete versions $\theta^*_{\min}$ and $\theta^*_{\max}$ of these optimisers are in the set of all allowed discrete values:

$$\theta^*_{\min} = \theta^*_{\max} \in \Theta^* \setminus \{0\}.$$

5.6.10 Negative Predictive Value

The *Negative Predictive Value* NPV_θ is the performance measure that indicates the fraction of all negatively predicted observations that are in fact negative. Hence, it shows how cautious the model is in assigning negative predictions. A large value means the model is cautious in predicting observations negatively, while a small value means the opposite.

Definition and Distribution

The *Negative Predictive Value* is commonly defined as

$$\mathrm{NPV}_\theta = \frac{\mathrm{TN}_\theta}{\mathrm{TN}_\theta + \mathrm{FN}_\theta}.$$

With the help of Equations (B3) and (B4), this definition can be rewritten as

$$\mathrm{NPV}_\theta = 1 - \frac{P - \mathrm{TP}_\theta}{M - \lfloor M \cdot \theta \rceil}.$$

Note that this performance measure is only defined whenever $\lfloor M \cdot \theta \rceil < M$, otherwise the denominator is zero. Therefore, we assume specifically for NPV_θ that $\theta < 1 - \frac{1}{2M}$. The definition of NPV_θ is linear in TP_θ and can thus be formulated as

$$\begin{aligned} \mathrm{NPV}_\theta &= Z_\theta\left(\frac{1}{M - \lfloor M \cdot \theta \rceil}, 1 - \frac{P}{M - \lfloor M \cdot \theta \rceil}\right) \\ &\sim H_\theta\left(\frac{1}{M - \lfloor M \cdot \theta \rceil}, 1 - \frac{P}{M - \lfloor M \cdot \theta \rceil}\right), \end{aligned} \tag{5.16}$$

with range:

$$\mathrm{NPV}_\theta \overset{(\mathrm{R})}{\in} \mathcal{R}\left(Z_\theta\left(\frac{1}{M - \lfloor M \cdot \theta \rceil}, 1 - \frac{P}{M - \lfloor M \cdot \theta \rceil}\right)\right).$$

Expectation

Since NPV_θ is linear in TP_θ with slope $a = 1/(M - \lfloor M \cdot \theta \rceil)$ and intercept $b = 1 - P/(M - \lfloor M \cdot \theta \rceil)$, its expectation is given by

$$\begin{aligned} \mathbf{E}[\mathrm{NPV}_\theta] &= \mathbf{E}\left[Z_\theta\left(\frac{1}{M - \lfloor M \cdot \theta \rceil}, 1 - \frac{P}{M - \lfloor M \cdot \theta \rceil}\right)\right] \\ &\overset{(\mathrm{E})}{=} \frac{1}{M - \lfloor M \cdot \theta \rceil} \cdot \mathbf{E}[\mathrm{TP}_\theta] + 1 - \frac{P}{M - \lfloor M \cdot \theta \rceil} = 1 - \frac{P}{M}. \end{aligned}$$

Optimal Baselines

The extreme values of the expectation of NPV_θ determine the baselines. They are given by

$$\min_{\theta \in [0,1-1/(2M))} (\mathbf{E}[\text{NPV}_\theta]) = 1 - \frac{P}{M},$$
$$\max_{\theta \in [0,1-1/(2M))} (\mathbf{E}[\text{NPV}_\theta]) = 1 - \frac{P}{M},$$

because the expectation does not depend on θ. Consequently, the optimisation values θ_{min} and θ_{max} are all allowed values for θ:

$$\theta_{\text{min}} = \theta_{\text{max}} \in \left[0, 1 - \frac{1}{2M}\right).$$

This also means the discrete forms θ^*_{min} and θ^*_{max} of the optimisers are in the set of all allowed discrete values:

$$\theta^*_{\text{min}} = \theta^*_{\text{max}} \in \Theta^* \setminus \{1\}.$$

5.6.11 False Discovery Rate

The *False Discovery Rate* FDR_θ is the performance measure that looks at the fraction of positively predicted observations that are actually negative. Therefore, it can be seen as the counterpart to the Positive Predictive Value that we discuss in Section 5.6.9. Consequently, a small value means the model is cautious in predicting observations as positive, while a large value means the opposite.

Definition and Distribution

The *False Discovery Rate* is commonly defined as

$$\text{FDR}_\theta = \frac{\text{FP}_\theta}{\text{TP}_\theta + \text{FP}_\theta} = 1 - \text{PPV}_\theta.$$

With the help of Equation (5.15), this definition can be rewritten as

$$\text{FDR}_\theta = 1 - \frac{\text{TP}_\theta}{\lfloor M \cdot \theta \rceil}.$$

Note that this performance measure is only defined whenever $\lfloor M \cdot \theta \rceil > 0$, otherwise the denominator is zero. Therefore, we assume specifically for FDR_θ that $\theta > \frac{1}{2M}$. The definition of FDR_θ is linear in TP_θ and can thus be formulated as

$$\text{FDR}_\theta = Z_\theta\left(-\frac{1}{\lfloor M \cdot \theta \rceil}, 1\right) \sim H_\theta\left(-\frac{1}{\lfloor M \cdot \theta \rceil}, 1\right),$$

with range:

$$\text{FDR}_\theta \overset{(\text{R})}{\in} \mathcal{R}\left(Z_\theta\left(-\frac{1}{\lfloor M \cdot \theta \rceil}, 1\right)\right).$$

Expectation

Since FDR_θ is linear in TP_θ with slope $a = -1/\lfloor M \cdot \theta \rceil$ and intercept $b = 1$, its expectation is given by

$$\mathbf{E}[\mathrm{FDR}_\theta] = \mathbf{E}\left[Z_\theta \left(-\frac{1}{\lfloor M \cdot \theta \rceil}, 1 \right) \right] \overset{(\mathrm{E})}{=} -\frac{1}{\lfloor M \cdot \theta \rceil} \cdot \mathbf{E}[\mathrm{TP}_\theta] + 1 = 1 - \frac{P}{M}.$$

Optimal Baselines

The extreme values of the expectation of FDR_θ determine the baselines. Its range is given by

$$\min_{\theta \in (1/(2M), 1]} (\mathbf{E}[\mathrm{FDR}_\theta]) = 1 - \frac{P}{M},$$
$$\max_{\theta \in (1/(2M), 1]} (\mathbf{E}[\mathrm{FDR}_\theta]) = 1 - \frac{P}{M},$$

because the expectation does not depend on θ. Consequently, the optimisation values $\theta_{\min}$ and $\theta_{\max}$ are all allowed values for θ:

$$\theta_{\min} = \theta_{\max} \in \left(\frac{1}{2M}, 1 \right].$$

This also means the discrete forms $\theta^*_{\min}$ and $\theta^*_{\max}$ of the optimisers are in the set of all allowed discrete values:

$$\theta^*_{\min} = \theta^*_{\max} \in \Theta^* \setminus \{0\}.$$

5.6.12 False Omission Rate

The *False Omission Rate* FOR_θ is the performance measure that considers the fraction of observations that are predicted negative, but are in fact positive. Hence, it can be seen as the counterpart to the Negative Predictive Value that is introduced in Section 5.6.10. As a consequence, a small value means the model is cautious is predicting observations negatively, while a large value means the opposite.

Definition and Distribution

The *False Omission Rate* is commonly defined as

$$\mathrm{FOR}_\theta = \frac{\mathrm{FN}_\theta}{\mathrm{TN}_\theta + \mathrm{FN}_\theta}.$$

With the aid of Equation (B3), this can be reformulated to

$$\mathrm{FOR}_\theta = \frac{P - \mathrm{TP}_\theta}{M - \lfloor M \cdot \theta \rceil}.$$

Note that this performance measure is only defined whenever $\lfloor M \cdot \theta \rceil < M$, otherwise the denominator is zero. Therefore, we assume specifically for FOR_θ that $\theta < 1 - \frac{1}{2M}$. Now, FOR_θ is linear in TP_θ and can therefore be written as

$$\begin{aligned}\mathrm{FOR}_\theta &= Z_\theta\left(-\frac{1}{M - \lfloor M \cdot \theta \rceil}, \frac{P}{M - \lfloor M \cdot \theta \rceil}\right)\\ &\sim H_\theta\left(-\frac{1}{M - \lfloor M \cdot \theta \rceil}, \frac{P}{M - \lfloor M \cdot \theta \rceil}\right),\end{aligned}$$

with range:

$$\mathrm{FOR}_\theta \overset{(\mathrm{R})}{\in} \mathcal{R}\left(Z_\theta\left(-\frac{1}{M - \lfloor M \cdot \theta \rceil}, \frac{P}{M - \lfloor M \cdot \theta \rceil}\right)\right).$$

Expectation

Because FOR_θ is linear in TP_θ with slope $a = -1/(M - \lfloor M \cdot \theta \rceil)$ and intercept $b = P/(M - \lfloor M \cdot \theta \rceil)$, its expectation is

$$\begin{aligned}\mathbf{E}[\mathrm{FOR}_\theta] &= \mathbf{E}\left[Z_\theta\left(-\frac{1}{M - \lfloor M \cdot \theta \rceil}, \frac{P}{M - \lfloor M \cdot \theta \rceil}\right)\right]\\ &\overset{(\mathrm{E})}{=} -\frac{1}{M - \lfloor M \cdot \theta \rceil} \cdot \mathbf{E}[\mathrm{TP}_\theta] + \frac{P}{M - \lfloor M \cdot \theta \rceil} = \frac{P}{M}.\end{aligned}$$

Optimal Baselines

The range of the expectation of FOR_θ determines the baselines. The extreme values are defined as

$$\begin{aligned}\min_{\theta \in [0, 1 - 1/(2M))} \left(\mathbf{E}[\mathrm{FOR}_\theta]\right) &= \frac{P}{M},\\ \max_{\theta \in [0, 1 - 1/(2M))} \left(\mathbf{E}[\mathrm{FOR}_\theta]\right) &= \frac{P}{M},\end{aligned}$$

because the expectation does not depend on θ. Consequently, the optimisation values $\theta_{\min}$ and $\theta_{\max}$ are all allowed values for θ:

$$\theta_{\min} = \theta_{\max} \in \left[0, 1 - \frac{1}{2M}\right).$$

This also means the discrete forms $\theta^*_{\min}$ and $\theta^*_{\max}$ of the optimisers are in the set of all allowed discrete values:

$$\theta^*_{\min} = \theta^*_{\max} \in \Theta^* \setminus \{1\}.$$

5.6.13 F_β Score

The F_β *score* $\mathrm{F}_\theta^{(\beta)}$ was introduced by Chinchor [16] in 1992. It is the weighted harmonic average between the True Positive Rate (TPR_θ) and the Positive Predictive Value (PPV_θ). These two performance measures are discussed extensively in Sections 5.6.5 and 5.6.9, respectively, and their summarised results are shown in Tables 5.3 and 5.4. The F_β score balances predicting the actual positive observations correctly (TPR_θ) and being cautious in predicting observations as positive (PPV_θ). The factor $\beta > 0$ indicates how much more TPR_θ is weighted compared to PPV_θ.

Definition and Distribution

The F_β score is commonly defined as

$$\mathrm{F}_\theta^{(\beta)} = \frac{1+\beta^2}{\frac{1}{\mathrm{PPV}_\theta} + \frac{\beta^2}{\mathrm{TPR}_\theta}}.$$

By using the definitions of TPR_θ and PPV_θ in Equations (5.10) and (5.14), $\mathrm{F}_\theta^{(\beta)}$ can be formulated in terms of the base measures:

$$\mathrm{F}_\theta^{(\beta)} = \frac{(1+\beta^2)\cdot \mathrm{TP}_\theta}{\beta^2 \cdot P + \mathrm{TP}_\theta + \mathrm{FP}_\theta}$$

Equations (B1) and (B2) allow us to write the formulation above in terms of only TP_θ:

$$\mathrm{F}_\theta^{(\beta)} = \frac{(1+\beta^2)\cdot \mathrm{TP}_\theta}{\beta^2 \cdot P + \lfloor M \cdot \theta \rceil}.$$

Note that $P > 0$ and $\lfloor M \cdot \theta \rceil > 0$, otherwise TPR_θ or PPV_θ is not defined, and hence, $\mathrm{F}_\theta^{(\beta)}$ is not defined. Now, $\mathrm{F}_\theta^{(\beta)}$ is linear in TP_θ and can be formulated as

$$\mathrm{F}_\theta^{(\beta)} = Z_\theta\left(\frac{1+\beta^2}{\beta^2 \cdot P + \lfloor M \cdot \theta \rceil}, 0\right),$$

with range:

$$\mathrm{F}_\theta^{(\beta)} \overset{(\mathrm{R})}{\in} \mathcal{R}\left(Z_\theta\left(\frac{1+\beta^2}{\beta^2 \cdot P + \lfloor M \cdot \theta \rceil}, 0\right)\right).$$

Expectation

Because $\mathrm{F}_\theta^{(\beta)}$ is linear in TP_θ with slope $a = (1+\beta^2)/(\beta^2 P + \lfloor M\cdot\theta\rceil)$ and intercept $b = 0$, its expectation is given by

$$\begin{aligned}\mathbf{E}[\mathrm{F}_\theta^{(\beta)}] = \mathbf{E}\left[Z_\theta\left(\frac{1+\beta^2}{\beta^2\cdot P + \lfloor M\cdot\theta\rceil}, 0\right)\right] &\overset{(\mathrm{E})}{=} \frac{1+\beta^2}{\beta^2\cdot P + \lfloor M\cdot\theta\rceil}\cdot\mathbf{E}[\mathrm{TP}_\theta] + 0\\ &= \frac{\lfloor M\cdot\theta\rceil\cdot P\cdot(1+\beta^2)}{M\cdot(\beta^2\cdot P+\lfloor M\cdot\theta\rceil)}\\ &= \frac{(1+\beta^2)\cdot P\cdot\theta^*}{\beta^2\cdot P + M\cdot\theta^*}. \end{aligned} \tag{5.17}$$

Optimal Baselines

To determine the extreme values of the expectation of $\mathrm{F}_\theta^{(\beta)}$, and therefore the baselines, the derivative of the function $f : [0,1] \to [0,1]$ defined as

$$f(t) = \frac{(1+\beta^2)\cdot P\cdot t}{\beta^2\cdot P + M\cdot t}$$

is calculated. First note that $\mathbf{E}[\mathrm{F}_\theta^{(\beta)}] = f(\lfloor M\cdot\theta\rceil/M)$. The derivative is given by

$$\frac{\mathrm{d}f(t)}{\mathrm{d}t} = \frac{\beta^2(1+\beta^2)\cdot P^2}{(\beta^2\cdot P+M\cdot t)^2}.$$

It is strictly positive for all t in its domain, thus f is strictly increasing in t. This means $\mathbf{E}[\mathrm{F}_\theta^{(\beta)}]$ given in Equation (5.17) is non-decreasing in both θ and θ^*. This is because the term $\lfloor M\cdot\theta\rceil/M$ is non-decreasing in θ. Hence, the extreme values of the expectation of $\mathrm{F}_\theta^{(\beta)}$ are its border values:

$$\begin{aligned}\min_{\theta\in[1/(2M),1]}\left(\mathbf{E}[\mathrm{F}_\theta^{(\beta)}]\right) &= \min_{\theta\in[1/(2M),1]}\left(\frac{(1+\beta^2)\cdot P\cdot\lfloor M\cdot\theta\rceil}{M(\beta^2\cdot P+\lfloor M\cdot\theta\rceil)}\right) = \frac{(1+\beta^2)\cdot P}{M(\beta^2\cdot P+1)},\\ \max_{\theta\in[1/(2M),1]}\left(\mathbf{E}[\mathrm{F}_\theta^{(\beta)}]\right) &= \max_{\theta\in[1/(2M),1]}\left(\frac{(1+\beta^2)\cdot P\cdot\lfloor M\cdot\theta\rceil}{M(\beta^2\cdot P+\lfloor M\cdot\theta\rceil)}\right) = \frac{(1+\beta^2)\cdot P}{\beta^2\cdot P+M}.\end{aligned}$$

Consequently, the optimisation values $\theta_{\min}$ and $\theta_{\max}$ for the extreme values are given by

$$\begin{aligned}\theta_{\min} \in \operatorname*{arg\,min}_{\theta\in[1/(2M),1]}\left(\mathbf{E}[\mathrm{F}_\theta^{(\beta)}]\right) &= \operatorname*{arg\,min}_{\theta\in[1/(2M),1]}\left(\frac{\lfloor M\cdot\theta\rceil}{\beta^2\cdot P+\lfloor M\cdot\theta\rceil}\right)\\ &= \begin{cases}[\frac{1}{2},1] & \text{if } M=1\\ [\frac{1}{2M},\frac{3}{2M}) & \text{if } M>1,\end{cases}\\ \theta_{\max} \in \operatorname*{arg\,max}_{\theta\in[1/(2M),1]}\left(\mathbf{E}[\mathrm{F}_\theta^{(\beta)}]\right) &= \operatorname*{arg\,max}_{\theta\in[1/(2M),1]}\left(\frac{\lfloor M\cdot\theta\rceil}{\beta^2\cdot P+\lfloor M\cdot\theta\rceil}\right)\\ &= \begin{cases}[\frac{1}{2},1] & \text{if } M=1\\ [1-\frac{1}{2M},1] & \text{if } M>1,\end{cases}\end{aligned}$$

respectively. Following this reasoning, the discrete forms θ^*_{min} and θ^*_{max} are given by

$$\theta^*_{\text{min}} \in \underset{\theta^* \in \Theta^* \setminus \{0\}}{\arg\min} \left\{ \mathbf{E}[\mathrm{F}^{(\beta)}_{\theta^*}] \right\} = \underset{\theta^* \in \Theta^* \setminus \{0\}}{\arg\min} \left\{ \frac{\theta^*}{\beta^2 \cdot P + M \cdot \theta^*} \right\} = \left\{ \frac{1}{M} \right\},$$

$$\theta^*_{\text{max}} \in \underset{\theta^* \in \Theta^* \setminus \{0\}}{\arg\max} \left\{ \mathbf{E}[\mathrm{F}^{(\beta)}_{\theta^*}] \right\} = \underset{\theta^* \in \Theta^* \setminus \{0\}}{\arg\max} \left\{ \frac{\theta^*}{\beta^2 \cdot P + M \cdot \theta^*} \right\} = \{1\}.$$

5.6.14 Youden's J Statistic

The *Youden's J Statistic* J_θ, *Youden's Index*, or *(Bookmaker) Informedness* was introduced by Youden [129] in 1950 to capture the performance of a diagnostic test as a single statistic. It incorporates both the True Positive Rate and the True Negative Rate, which are discussed in Sections 5.6.5 and 5.6.8, respectively. Youden's J Statistic shows how well the model is able to correctly predict both the positive as the negative observations.

Definition and Distribution

The Youden's J Statistic is commonly defined as

$$\mathrm{J}_\theta = \mathrm{TPR}_\theta + \mathrm{TNR}_\theta - 1.$$

By using Equations (5.11) and (5.13), which provide the definitions of TPR_θ and TNR_θ in terms of TP_θ, the definition of J_θ can be reformulated as

$$\mathrm{J}_\theta = \frac{M \cdot \mathrm{TP}_\theta - P \cdot \lfloor M \cdot \theta \rceil}{P\,(M - P)}.$$

Because TPR_θ needs $P > 0$, and TNR_θ needs $N > 0$, we have both these assumptions for J_θ. Consequently, $M > 1$. Now, J_θ is linear in TP_θ and can therefore be written as

$$\mathrm{J}_\theta = Z_\theta \left(\frac{M}{P\,(M-P)}, -\frac{\lfloor M \cdot \theta \rceil}{M - P} \right) \sim H_\theta \left(\frac{M}{P\,(M-P)}, -\frac{\lfloor M \cdot \theta \rceil}{M - P} \right),$$

with range:

$$\mathrm{J}_\theta \overset{(\mathrm{R})}{\in} \mathcal{R} \left(Z_\theta \left(\frac{M}{P\,(M-P)}, -\frac{\lfloor M \cdot \theta \rceil}{M - P} \right) \right).$$

Expectation

Since J_θ is linear in TP_θ with slope $a = M/(P\,(M-P))$ and intercept $b = -\lfloor M \cdot \theta \rceil / (M - P)$, its expectation is given by

$$\begin{aligned}\mathbf{E}[\mathrm{J}_\theta] &= \mathbf{E}\left[Z_\theta \left(\frac{M}{P\,(M-P)}, -\frac{\lfloor M \cdot \theta \rceil}{M - P} \right) \right] \overset{(\mathrm{E})}{=} \frac{M}{P\,(M-P)} \cdot \mathbf{E}[\mathrm{TP}_\theta] - \frac{\lfloor M \cdot \theta \rceil}{M - P} \\ &= 0.\end{aligned}$$

Optimal Baselines

The extreme values of the expectation of J_θ determine the baselines. They are given by

$$\min_{\theta \in [0,1]} (\mathbf{E}[\mathrm{J}_\theta]) = 0,$$
$$\max_{\theta \in [0,1]} (\mathbf{E}[\mathrm{J}_\theta]) = 0,$$

because the expected value does not depend on θ. Consequently, the optimisation values θ_{min} and θ_{max} can be any value in the domain of θ:

$$\theta_{\text{min}} = \theta_{\text{max}} \in [0,1].$$

This also holds for the discrete forms θ^*_{min} and θ^*_{max} of the optimisers:

$$\theta^*_{\text{min}} = \theta^*_{\text{max}} \in \Theta^*.$$

5.6.15 Markedness

The *Markedness* MK_θ or *deltaP* is a performance measure that is mostly used in linguistics and social sciences. It combines both the Positive Predictive Value and the Negative Predictive Value. These two measures are discussed in Sections 5.6.9 and 5.6.10, respectively. The Markedness indicates how cautious the model is in predicting observations as positive and also how cautious it is in predicting them as negative.

Definition and Distribution

The Markedness is commonly defined as

$$\mathrm{MK}_\theta = \mathrm{PPV}_\theta + \mathrm{NPV}_\theta - 1.$$

This definition of MK_θ can be reformulated in terms of TP_θ by using Equations (5.15) and (5.16):

$$\mathrm{MK}_\theta = \frac{M \cdot \mathrm{TP}_\theta - P \cdot \lfloor M \cdot \theta \rceil}{\lfloor M \cdot \theta \rceil (M - \lfloor M \cdot \theta \rceil)}.$$

Note that MK_θ is only defined for $M > 1$ and $\theta \in [1/(2M), 1 - 1/(2M))$, otherwise the denominator becomes zero. The assumption $M > 1$ automatically follows from the assumptions $\hat{P} > 0$ and $\hat{N} > 0$, which hold for PPV_θ and NPV_θ, respectively. In other words, there is at least one observation predicted positive and at least one predicted negative, thus $M > 1$. Now, MK_θ is linear in TP_θ and can therefore be written as

$$\begin{aligned}\mathrm{MK}_\theta &= Z_\theta \left(\frac{M}{\lfloor M \cdot \theta \rceil (M - \lfloor M \cdot \theta \rceil)}, - \frac{P}{M - \lfloor M \cdot \theta \rceil} \right) \\ &\sim H_\theta \left(\frac{M}{\lfloor M \cdot \theta \rceil (M - \lfloor M \cdot \theta \rceil)}, - \frac{P}{M - \lfloor M \cdot \theta \rceil} \right),\end{aligned}$$

with range:

$$\mathrm{MK}_\theta \overset{(\mathrm{R})}{\in} \mathcal{R}\left(Z_\theta\left(\frac{M}{\lfloor M\cdot\theta\rceil(M-\lfloor M\cdot\theta\rceil)}, -\frac{P}{M-\lfloor M\cdot\theta\rceil}\right)\right).$$

Expectation

By using slope $a = M/(\lfloor M\cdot\theta\rceil(M-\lfloor M\cdot\theta\rceil))$ and intercept $b = -P/(M-\lfloor M\cdot\theta\rceil)$, the expectation of MK_θ can be calculated:

$$\begin{aligned}\mathbf{E}[\mathrm{MK}_\theta] &= \mathbf{E}\left[Z_\theta\left(\frac{M}{\lfloor M\cdot\theta\rceil(M-\lfloor M\cdot\theta\rceil)}, -\frac{P}{M-\lfloor M\cdot\theta\rceil}\right)\right]\\ &\overset{(\mathrm{E})}{=} \frac{M}{\lfloor M\cdot\theta\rceil(M-\lfloor M\cdot\theta\rceil)}\cdot\mathbf{E}[\mathrm{TP}_\theta] - \frac{P}{M-\lfloor M\cdot\theta\rceil} = 0.\end{aligned}$$

Optimal Baselines

The extreme values of the expectation of MK_θ determine the baselines. Its range is given by:

$$\begin{aligned}\min_{\theta\in[1/(2M),1-1/(2M))}(\mathbf{E}[\mathrm{MK}_\theta]) &= 0,\\ \max_{\theta\in[1/(2M),1-1/(2M))}(\mathbf{E}[\mathrm{MK}_\theta]) &= 0,\end{aligned}$$

since the expected value does not depend on θ. Therefore, the optimisation values $\theta_{\min}$ and $\theta_{\max}$ are in the set of allowed values for θ:

$$\theta_{\min} = \theta_{\max} \in \left[\frac{1}{2M}, 1-\frac{1}{2M}\right).$$

This also means the discrete forms $\theta^*_{\min}$ and $\theta^*_{\max}$ of the optimisers are in the set of the allowed discrete values:

$$\theta^*_{\min} = \theta^*_{\max} \in \Theta^* \setminus \{0,1\}.$$

5.6.16 Accuracy

The *Accuracy* Acc_θ is the performance measure that assesses how good the model is in correctly predicting the observations without making a distinction between positive or negative observations.

Definition and Distribution

The Accuracy is commonly defined as

$$\mathrm{Acc}_\theta = \frac{\mathrm{TP}_\theta + \mathrm{TN}_\theta}{M}.$$

By using Equation (B4), this can be restated as

$$\text{Acc}_\theta = \frac{2 \cdot \text{TP}_\theta + M - P - \lfloor M \cdot \theta \rceil}{M}.$$

Note that it is linear in TP_θ and can therefore be written as

$$\text{Acc}_\theta = Z_\theta\left(\frac{2}{M}, \frac{M - P - \lfloor M \cdot \theta \rceil}{M}\right) \sim H_\theta\left(\frac{2}{M}, \frac{M - P - \lfloor M \cdot \theta \rceil}{M}\right), \tag{5.18}$$

with range:

$$\text{Acc}_\theta \overset{(R)}{\in} \mathcal{R}\left(Z_\theta\left(\frac{2}{M}, \frac{M - P - \lfloor M \cdot \theta \rceil}{M}\right)\right).$$

Expectation

Since Acc_θ is linear in TP_θ with slope $a = 2/M$ and intercept $b = (M - P - \lfloor M \cdot \theta \rceil)/M$, its expectation can be derived:

$$\begin{aligned}
\mathbf{E}[\text{Acc}_\theta] &= \mathbf{E}\left[Z_\theta\left(\frac{2}{M}, \frac{M - P - \lfloor M \cdot \theta \rceil}{M}\right)\right] \\
&\overset{(E)}{=} \frac{2}{M} \cdot \mathbf{E}[\text{TP}_\theta] + \frac{M - P - \lfloor M \cdot \theta \rceil}{M} \\
&= \frac{(M - \lfloor M \cdot \theta \rceil)(M - P) + \lfloor M \cdot \theta \rceil \cdot P}{M^2} \\
&= \frac{(1 - \theta^*)(M - P) + \theta^* \cdot P}{M}.
\end{aligned} \tag{5.19}$$

Optimal Baselines

The range of the expectation of Acc_θ directly determines the baselines. To determine the extreme values of Acc_θ, the derivative of the function $f : [0, 1] \to [0, 1]$ defined as

$$f(t) = \frac{(1 - t)(M - P) + P \cdot t}{M}$$

is calculated. First, note that $\mathbf{E}[\text{Acc}_\theta] = f(\lfloor M \cdot \theta \rceil / M)$. The derivative is given by

$$\frac{\mathrm{d}f(t)}{\mathrm{d}t} = \frac{2P - M}{M}.$$

It does not depend on t, but whether the derivative is positive or negative depends on P and M. Whenever $P > \frac{M}{2}$, then f is strictly increasing for all t in its domain. If $P < \frac{M}{2}$, then f is strictly decreasing. When $P = \frac{M}{2}$, f is constant. Consequently, the same holds for $\mathbf{E}[\text{Acc}_\theta]$ given in Equation (5.19). This is because the term $\lfloor M \cdot \theta \rceil / M$ is non-decreasing in θ. Thus, the extreme values of the expectation of Acc_θ are given by

$$\min_{\theta\in[0,1]} (\mathbf{E}[\text{Acc}_\theta]) = \begin{cases} \frac{P}{M} & \text{if } P < \frac{M}{2} \\ 1-\frac{P}{M} & \text{if } P \geq \frac{M}{2} \end{cases} = \min\left\{\frac{P}{M}, 1-\frac{P}{M}\right\},$$

$$\max_{\theta\in[0,1]} (\mathbf{E}[\text{Acc}_\theta]) = \begin{cases} 1-\frac{P}{M} & \text{if } P < \frac{M}{2} \\ \frac{P}{M} & \text{if } P \geq \frac{M}{2} \end{cases} = \max\left\{\frac{P}{M}, 1-\frac{P}{M}\right\}.$$

This means that the optimisation values $\theta_{\min} \in [0,1]$ and $\theta_{\max} \in [0,1]$ for these extreme values respectively are given by

$$\theta_{\min} \in \arg\min_{\theta\in[0,1]} (\mathbf{E}[\text{Acc}_\theta]) = \begin{cases} \left[1-\frac{1}{2M}, 1\right] & \text{if } P < \frac{M}{2} \\ [0,1] & \text{if } P = \frac{M}{2} \\ \left[0, \frac{1}{2M}\right) & \text{if } P > \frac{M}{2}, \end{cases} \tag{5.20}$$

$$\theta_{\max} \in \arg\max_{\theta\in[0,1]} (\mathbf{E}[\text{Acc}_\theta]) = \begin{cases} \left[0, \frac{1}{2M}\right) & \text{if } P < \frac{M}{2} \\ [0,1] & \text{if } P = \frac{M}{2} \\ \left[1-\frac{1}{2M}, 1\right] & \text{if } P > \frac{M}{2}. \end{cases} \tag{5.21}$$

Consequently, the discrete versions $\theta^*_{\min} \in \Theta^*$ and $\theta^*_{\max} \in \Theta^*$ of the optimisers are given by

$$\theta^*_{\min} \in \arg\min_{\theta^*\in\Theta^*} \{\mathbf{E}[\text{Acc}_{\theta^*}]\} = \begin{cases} \{1\} & \text{if } P < \frac{M}{2} \\ \Theta^* & \text{if } P = \frac{M}{2} \\ \{0\} & \text{if } P > \frac{M}{2}, \end{cases} \tag{5.22}$$

$$\theta^*_{\max} \in \arg\max_{\theta^*\in\Theta^*} \{\mathbf{E}[\text{Acc}_{\theta^*}]\} = \begin{cases} \{0\} & \text{if } P < \frac{M}{2} \\ \Theta^* & \text{if } P = \frac{M}{2} \\ \{1\} & \text{if } P > \frac{M}{2}, \end{cases} \tag{5.23}$$

respectively.

5.6.17 Balanced Accuracy

The *Balanced Accuracy* BAcc_θ is the mean of the True Positive Rate and True Negative Rate, which are discussed in Sections 5.6.5 and 5.6.8. It determines how good the model is in correctly predicting the positive observations and in correctly predicting the negative observations on average.

Definition and Distribution

The Balanced Accuracy is commonly defined as

$$\text{BAcc}_\theta = \frac{1}{2} \cdot (\text{TPR}_\theta + \text{TNR}_\theta).$$

By using Equations (5.11) and (5.13), this can be reformulated as

$$\text{BAcc}_\theta = \frac{1}{2}\left(\frac{\text{TP}_\theta}{P} + 1 - \frac{\lfloor M\cdot\theta\rceil - \text{TP}_\theta}{M-P}\right) = \frac{M\cdot\text{TP}_\theta}{2P\,(M-P)} + \frac{M-P-\lfloor M\cdot\theta\rceil}{2\,(M-P)}.$$

Note that $P > 0$ and $N > 0$ should hold, otherwise TPR_θ or TNR_θ is not defined. Consequently, $M > 1$. Note that BAcc_θ is linear in TP_θ and can therefore be written as

$$\begin{aligned}\text{BAcc}_\theta &= Z_\theta\left(\frac{M}{2P\,(M-P)}, \frac{M-P-\lfloor M\cdot\theta\rceil}{2\,(M-P)}\right)\\ &\sim H_\theta\left(\frac{M}{2P\,(M-P)}, \frac{M-P-\lfloor M\cdot\theta\rceil}{2\,(M-P)}\right),\end{aligned}$$

with range:

$$\text{BAcc}_\theta \overset{(\text{R})}{\in} \mathcal{R}\left(Z_\theta\left(\frac{M}{2P\,(M-P)}, \frac{M-P-\lfloor M\cdot\theta\rceil}{2\,(M-P)}\right)\right).$$

Expectation

BAcc_θ is linear in TP_θ with slope $a = M/(2P\,(M-P))$ and intercept $b = (M-P-\lfloor M\cdot\theta\rceil)/(2\,(M-P))$, so its expectation can be derived:

$$\begin{aligned}\mathbf{E}[\text{BAcc}_\theta] &= \mathbf{E}\left[Z_\theta\left(\frac{M}{2P\,(M-P)}, \frac{M-P-\lfloor M\cdot\theta\rceil}{2\,(M-P)}\right)\right]\\ &\overset{(\text{E})}{=} \frac{M}{2P\,(M-P)}\cdot\mathbf{E}[\text{TP}_\theta] + \frac{M-P-\lfloor M\cdot\theta\rceil}{2\,(M-P)} = \frac{1}{2}.\end{aligned}$$

Optimal Baselines

The baselines are directly determined by the ranges of the expectation of BAcc_θ. Since the expectation is constant, its extreme values are the same:

$$\begin{aligned}\min_{\theta\in[0,1]}\left(\mathbf{E}[\text{BAcc}_\theta]\right) &= \frac{1}{2},\\ \max_{\theta\in[0,1]}\left(\mathbf{E}[\text{BAcc}_\theta]\right) &= \frac{1}{2}.\end{aligned}$$

This means that the optimisation values $\theta_{\min} \in [0,1]$ and $\theta_{\max} \in [0,1]$ for these extreme values respectively are simply

$$\begin{aligned}\theta_{\min} &\in \operatorname*{arg\,min}_{\theta\in[0,1]}\left(\mathbf{E}[\text{BAcc}_\theta]\right) = [0,1],\\ \theta_{\max} &\in \operatorname*{arg\,max}_{\theta\in[0,1]}\left(\mathbf{E}[\text{BAcc}_\theta]\right) = [0,1].\end{aligned}$$

Consequently, the discrete versions $\theta^*_{\min} \in \Theta^*$ and $\theta^*_{\max} \in \Theta^*$ of the optimisers are given by

$$\begin{aligned}\theta^*_{\min} &\in \operatorname*{arg\,min}_{\theta^*\in\Theta^*}\left\{\mathbf{E}[\text{Acc}_{\theta^*}]\right\} = \Theta^*,\\ \theta^*_{\max} &\in \operatorname*{arg\,max}_{\theta^*\in\Theta^*}\left\{\mathbf{E}[\text{Acc}_{\theta^*}]\right\} = \Theta^*,\end{aligned}$$

respectively.

5.6.18 Matthews Correlation Coefficient

The *Matthews Correlation Coefficient* MCC_θ was established by Matthews [70]. However, its definition is identical to that of the Yule phi coefficient, which was introduced by Yule [130]. The performance measure can be seen as the correlation coefficient between the actual and predicted classes. Hence, it is one of the few measures that lies in $[-1,1]$ instead of $[0,1]$.

Definition and Distribution

The Matthews Correlation Coefficient is commonly defined as

$$\mathrm{MCC}_\theta = \frac{\mathrm{TP}_\theta \cdot \mathrm{TN}_\theta - \mathrm{FN}_\theta \cdot \mathrm{FP}_\theta}{\sqrt{(\mathrm{TP}_\theta + \mathrm{FP}_\theta)(\mathrm{TP}_\theta + \mathrm{FN}_\theta)(\mathrm{TN}_\theta + \mathrm{FP}_\theta)(\mathrm{TN}_\theta + \mathrm{FN}_\theta)}}.$$

By using Equations (B2) and (B4), this definition can be reformulated as

$$\mathrm{MCC}_\theta = \frac{M \cdot \mathrm{TP}_\theta - P \cdot \lfloor M \cdot \theta \rceil}{\sqrt{\lfloor M \cdot \theta \rceil \cdot P\,(M-P)\,(M - \lfloor M \cdot \theta \rceil)}}. \tag{5.24}$$

As Table 5.2 shows, the assumptions $P > 0$, $N > 0$, $\hat{P} := \lfloor M \cdot \theta \rceil > 0$, and $\hat{N} := M - \lfloor M \cdot \theta \rceil > 0$ must hold. If one of these assumptions is violated, then the denominator in Equation (5.24) is zero, and MCC_θ is not defined. Therefore, we have for MCC_θ that $\frac{1}{2M} \leq \theta < 1 - \frac{1}{2M}$ and $M > 1$. Next, to improve readability we introduce the variable $C(M, P, \theta)$ to replace the denominator in Equation (5.24):

$$C(M,P,\theta) := \sqrt{\lfloor M \cdot \theta \rceil \cdot P\,(M-P)\,(M - \lfloor M \cdot \theta \rceil)}.$$

The definition of MCC_θ is linear in TP_θ and can thus be formulated as

$$\mathrm{MCC}_\theta = Z_\theta\left(\frac{M}{C(M,P,\theta)}, \frac{-P \cdot \lfloor M \cdot \theta \rceil}{C(M,P,\theta)}\right) \sim H_\theta\left(\frac{M}{C(M,P,\theta)}, \frac{-P \cdot \lfloor M \cdot \theta \rceil}{C(M,P,\theta)}\right),$$

with range:

$$\mathrm{MCC}_\theta \overset{(\mathrm{R})}{\in} \mathcal{R}\left(Z_\theta\left(\frac{M}{C(M,P,\theta)}, \frac{-P \cdot \lfloor M \cdot \theta \rceil}{C(M,P,\theta)}\right)\right).$$

Expectation

MCC_θ is linear in TP_θ with slope $a = M/C(M,P,\theta)$ and intercept $b = -P \cdot \lfloor M \cdot \theta \rceil / C(M,P,\theta)$, so its expectation can be derived from Equation (E):

$$\begin{aligned}\mathbf{E}[\mathrm{MCC}_\theta] &= \mathbf{E}\left[Z_\theta\left(\frac{M}{C(M,P,\theta)}, \frac{-P \cdot \lfloor M \cdot \theta \rceil}{C(M,P,\theta)}\right)\right] \\ &\overset{(\mathrm{E})}{=} \frac{M}{C(M,P,\theta)} \cdot \mathbf{E}[\mathrm{TP}_\theta] - \frac{P \cdot \lfloor M \cdot \theta \rceil}{C(M,P,\theta)} = 0.\end{aligned}$$

Optimal Baselines

The baselines are directly determined by the ranges of the expectation of MCC_θ. Since the expectation is constant, its extreme values are the same:

$$\min_{\theta \in [1/(2M), 1-1/(2M))} (\mathbf{E}[\mathrm{MCC}_\theta]) = 0,$$
$$\max_{\theta \in [1/(2M), 1-1/(2M))} (\mathbf{E}[\mathrm{MCC}_\theta]) = 0.$$

This means that the optimisation values $\theta_{\min}$ and $\theta_{\max}$ for these extreme values respectively are simply:

$$\theta_{\min} = \theta_{\max} \in \left[\frac{1}{2M}, 1 - \frac{1}{2M}\right).$$

Consequently, the discrete versions $\theta^*_{\min}$ and $\theta^*_{\max}$ of the optimisers are given by:

$$\theta^*_{\min} = \theta^*_{\max} \in \Theta^* \setminus \{0, 1\}.$$

5.6.19 Cohen's Kappa

Cohen's kappa κ_θ is a less straightforward performance measure than the other measures that we discuss in this research. It is used to quantify the inter-rater reliability for two raters of categorical observations [53]. In our case, we compare the first rater, which is the Dutch Draw classifier, with the perfect rater, which assigns the true label to each observation.

Definition and Distribution

Although there are several definitions for Cohen's kappa, here we choose the following:

$$\kappa_\theta = \frac{P_o^\theta - P_e^\theta}{1 - P_e^\theta},$$

with P_o^θ the Accuracy Acc_θ as defined in Section 5.6.16 and P_e^θ the probability that the shuffle approach assigns the true label by chance. These two values can be expressed in terms of the base measures as follows:

$$P_o^\theta = \mathrm{Acc}_\theta = \frac{\mathrm{TP}_\theta + \mathrm{TN}_\theta}{M},$$
$$P_e^\theta = \frac{(\mathrm{TP}_\theta + \mathrm{FP}_\theta) \cdot P + (\mathrm{TN}_\theta + \mathrm{FN}_\theta)(M - P)}{M^2}.$$

By using Equations (5.18), (B1), (B2), (B3) and (B4) the above can be rewritten as

$$P_o^\theta = \frac{2 \cdot \mathrm{TP}_\theta + M - P - \lfloor M \cdot \theta \rceil}{M},$$
$$P_e^\theta = \frac{\lfloor M \cdot \theta \rceil \cdot P + (M - \lfloor M \cdot \theta \rceil)(M - P)}{M^2}.$$

Note that for κ_θ to be well-defined, we need $1 - P_e^\theta \neq 0$. In other words,

$$\lfloor M \cdot \theta \rceil \cdot P + (M - \lfloor M \cdot \theta \rceil)(M - P) \neq M^2.$$

This simplifies to

$$\frac{\lfloor M \cdot \theta \rceil}{M} \neq \frac{P}{2P - M}. \tag{5.25}$$

The left-hand side is by definition in the interval $[0, 1]$. For the right-hand side to be in that interval, we firstly need $P/(2P-M) \geq 0$. Since $P \geq 0$, that means $2P - M > 0$, and hence, $P > \frac{M}{2}$. Secondly, $P/(2P - M) \leq 1$. Since we know $P > \frac{M}{2}$, we obtain $P \geq M$. This inequality reduces to $P = M$, because P is always at most M. Whenever $P = M$, then Equation (5.25) becomes

$$\frac{\lfloor M \cdot \theta \rceil}{M} \neq 1.$$

To summarise, when $P < M$, then all $\theta \in [0, 1]$ are allowed in κ_θ, but when $P = M$, then $\theta < 1 - 1/(2M)$.

Now, by using P_o^θ and P_e^θ in the definition of Cohen's kappa, we obtain:

$$\kappa_\theta = \frac{2 \cdot M \cdot \mathrm{TP}_\theta - 2 \cdot \lfloor M \cdot \theta \rceil \cdot P}{P(M - \lfloor M \cdot \theta \rceil) + (M - P)\lfloor M \cdot \theta \rceil}.$$

To improve readability, we introduce the variables a_{κ_θ} and b_{κ_θ} defined as

$$\begin{aligned} a_{\kappa_\theta} &= \frac{2M}{P(M - \lfloor M \cdot \theta \rceil) + (M - P)\lfloor M \cdot \theta \rceil} \\ b_{\kappa_\theta} &= -\frac{2 \cdot \lfloor M \cdot \theta \rceil \cdot P}{P(M - \lfloor M \cdot \theta \rceil) + (M - P)\lfloor M \cdot \theta \rceil}. \end{aligned}$$

Hence, κ_θ is linear in TP_θ and can be written as

$$\kappa_\theta = Z_\theta(a_{\kappa_\theta}, b_{\kappa_\theta}) \sim H_\theta(a_{\kappa_\theta}, b_{\kappa_\theta}),$$

with range:

$$\kappa_\theta \overset{(\mathrm{R})}{\in} \mathcal{R}(Z_\theta(a_{\kappa_\theta}, b_{\kappa_\theta})).$$

Expectation

As Cohen's kappa is linear in TP_θ, its expectation can be derived:

$$\begin{aligned} \mathbf{E}[\kappa_\theta] = \mathbf{E}[Z_\theta(a_{\kappa_\theta}, b_{\kappa_\theta})] &\overset{(\mathrm{E})}{=} a_{\kappa_\theta} \cdot \mathbf{E}[\mathrm{TP}_\theta] + b_{\kappa_\theta} \\ &= \frac{2 \cdot \lfloor M \cdot \theta \rceil \cdot P}{P(M - \lfloor M \cdot \theta \rceil) + (M - P)\lfloor M \cdot \theta \rceil} \\ &\quad - \frac{2 \cdot \lfloor M \cdot \theta \rceil \cdot P}{P(M - \lfloor M \cdot \theta \rceil) + (M - P)\lfloor M \cdot \theta \rceil} \\ &= 0. \end{aligned}$$

Optimal Baselines

The baselines are directly determined by the ranges of the expectation of κ_θ. Since the expectation is constant, its extreme values are the same:

$$\begin{cases} \min_{\theta \in [0,1]} \left(\mathbf{E}[\kappa_\theta]\right) = 0 & \text{if } P < M \\ \min_{\theta \in [0,1-1/(2M))} \left(\mathbf{E}[\kappa_\theta]\right) = 0 & \text{if } P = M, \end{cases}$$

$$\begin{cases} \max_{\theta \in [0,1]} \left(\mathbf{E}[\kappa_\theta]\right) = 0 & \text{if } P < M \\ \max_{\theta \in [0,1-1/(2M))} \left(\mathbf{E}[\kappa_\theta]\right) = 0 & \text{if } P = M. \end{cases}$$

This means that the optimisation values $\theta_{\min}$ and $\theta_{\max}$ for these extreme values respectively are simply all allowed values:

$$\begin{cases} \theta_{\min} = \theta_{\max} \in [0,1] & \text{if } P < M \\ \theta_{\min} = \theta_{\max} \in \left[0, 1 - \frac{1}{2M}\right] & \text{if } P = M. \end{cases}$$

Consequently, the discrete versions $\theta^*_{\min}$ and $\theta^*_{\max}$ of the optimisers are given by

$$\begin{cases} \theta^*_{\min} = \theta^*_{\max} \in \Theta^* & \text{if } P < M \\ \theta^*_{\min} = \theta^*_{\max} \in \Theta^* \setminus \{1\} & \text{if } P = M. \end{cases}$$

5.6.20 Fowlkes-Mallows Index

The *Fowlkes-Mallows Index* FM_θ or *G-mean 1* was introduced by Fowlkes and Mallows in 1983 as a way to calculate the similarity between two clusterings [29]. It is the geometric average between the True Positive Rate (TPR_θ) and Positive Predictive Value (PPV_θ), which are discussed in Sections 5.6.5 and 5.6.9, respectively. It offers a balance between correctly predicting the actual positive observations (TPR_θ) and being cautious in predicting observations as positive (PPV_θ).

Definition and Distribution

The Fowlkes-Mallows Index is commonly defined as

$$\text{FM}_\theta = \sqrt{\text{TPR}_\theta \cdot \text{PPV}_\theta}.$$

By using the definitions of TPR_θ and PPV_θ in terms of TP_θ in, respectively, Equations (5.11) and (5.15), we obtain:

$$\text{FM}_\theta = \frac{\text{TP}_\theta}{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}}.$$

Since TPR_θ is only defined when $P > 0$ and PPV_θ only when $\hat{P} := \lfloor M \cdot \theta \rceil > 0$, also FM_θ has these assumptions. Therefore, $\theta \geq \frac{1}{2M}$. The definition of FM_θ is linear in TP_θ and can thus be formulated as

$$\mathrm{FM}_\theta = Z_\theta \left(\frac{1}{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}}, 0 \right) \sim H_\theta \left(\frac{1}{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}}, 0 \right),$$

with range:

$$\mathrm{FM}_\theta \overset{(\mathrm{R})}{\in} \mathcal{R} \left(Z_\theta \left(\frac{1}{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}}, 0 \right) \right).$$

Expectation

Because FM_θ is linear in TP_θ with slope $a = 1/\sqrt{P \cdot \lfloor M \cdot \theta \rceil}$ and intercept $b = 0$, its expectation is

$$\begin{aligned}\mathbf{E}[\mathrm{FM}_\theta] &= \mathbf{E}\left[Z_\theta \left(\frac{1}{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}}, 0 \right) \right] \overset{(\mathrm{E})}{=} \frac{1}{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}} \cdot \mathbf{E}[\mathrm{TP}_\theta] + 0 \\ &= \frac{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}}{M} = \sqrt{\frac{\theta^* \cdot P}{M}}.\end{aligned}$$

Optimal Baselines

The extreme values of the expectation of FM_θ determine the baselines. They are given by:

$$\min_{\theta \in [1/(2M),1]} (\mathbf{E}[\mathrm{FM}_\theta]) = \min_{\theta \in [1/(2M),1]} \left(\frac{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}}{M} \right) = \frac{\sqrt{P}}{M},$$

$$\max_{\theta \in [1/(2M),1]} (\mathbf{E}[\mathrm{FM}_\theta]) = \max_{\theta \in [1/(2M),1]} \left(\frac{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}}{M} \right) = \sqrt{\frac{P}{M}},$$

because the expectation is a non-decreasing function in θ. Note that the minimum and maximum are equal to each other when $M = 1$. Consequently, the optimisers $\theta_{\min}$ and $\theta_{\max}$ for the extreme values are determined by:

$$\begin{aligned}\theta_{\min} &\in \operatorname*{arg\,min}_{\theta \in [1/(2M),1]} (\mathbf{E}[\mathrm{FM}_\theta]) = \operatorname*{arg\,min}_{\theta \in [1/(2M),1]} \left(\frac{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}}{M} \right) \\ &= \begin{cases} \left[\frac{1}{2M}, 1\right] & \text{if } M = 1 \\ \left[\frac{1}{2M}, \frac{3}{2M}\right) & \text{if } M > 1, \end{cases}\end{aligned}$$

$$\begin{aligned}\theta_{\max} &\in \operatorname*{arg\,max}_{\theta \in [1/(2M),1]} (\mathbf{E}[\mathrm{FM}_\theta]) = \operatorname*{arg\,max}_{\theta \in [1/(2M),1]} \left(\frac{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}}{M} \right) \\ &= \begin{cases} \left[\frac{1}{2M}, 1\right] & \text{if } M = 1 \\ \left[1 - \frac{1}{2M}, 1\right] & \text{if } M > 1, \end{cases}\end{aligned}$$

respectively. The discrete forms θ^*_{min} and θ^*_{max} of these are given by:

$$\theta^*_{\text{min}} \in \underset{\theta^* \in \Theta^* \setminus \{0\}}{\arg\min} \{\mathbf{E}[\text{FM}_{\theta^*}]\} = \underset{\theta^* \in \Theta^* \setminus \{0\}}{\arg\min} \left\{ \sqrt{\frac{\theta^* \cdot P}{M}} \right\} = \left\{ \frac{1}{M} \right\},$$

$$\theta^*_{\text{max}} \in \underset{\theta^* \in \Theta^* \setminus \{0\}}{\arg\max} \{\mathbf{E}[\text{FM}_{\theta^*}]\} = \underset{\theta^* \in \Theta^* \setminus \{0\}}{\arg\max} \left\{ \sqrt{\frac{\theta^* \cdot P}{M}} \right\} = \{1\}.$$

5.6.21 G-mean 2

The *G-mean 2* $\text{G}^{(2)}_\theta$ was established by [50]. This performance measure is the geometric average between the True Positive Rate (TPR_θ) and True Negative Rate (TNR_θ), which we discuss in Sections 5.6.5 and 5.6.8, respectively. Hence, it balances correctly predicting the positive observations and correctly predicting the negative observations.

Definition and Distribution

The G-mean 2 is defined as

$$\text{G}^{(2)}_\theta = \sqrt{\text{TPR}_\theta \cdot \text{TNR}_\theta}.$$

Since TPR_θ needs the assumption $P > 0$ and TNR_θ needs $N := M - P > 0$, we have these restrictions also for $\text{G}^{(2)}_\theta$. Consequently, $M > 1$. Now, by using the definitions of TPR_θ and TNR_θ in terms of TP_θ in, respectively, Equations (5.11) and (5.13), we obtain:

$$\text{G}^{(2)}_\theta = \sqrt{\frac{\text{TP}_\theta \cdot (M - P - \lfloor M \cdot \theta \rceil) + \text{TP}^2_\theta}{P\,(M - P)}}.$$

This function is not a linear function of TP_θ, and hence, we cannot write it in the form $Z_\theta\,(a, b) = a \cdot \text{TP}_\theta + b$ for some variables $a, b \in \mathbb{R}$.

Expectation

Since $\text{G}^{(2)}_\theta$ is not linear in TP_θ, we cannot easily use the expectation of TP_θ to determine that for $\text{G}^{(2)}_\theta$. However, we are able to determine the second moment of $\mathrm{G}_\theta^{(2)}$:

$$\begin{aligned}
\mathbf{E}\left[\left(\mathrm{G}_\theta^{(2)}\right)^2\right] &= \frac{M-P-\lfloor M\cdot\theta\rceil}{P\,(M-P)}\cdot\mathbf{E}[\mathrm{TP}_\theta] + \frac{1}{P\,(M-P)}\cdot\mathbf{E}[\mathrm{TP}_\theta^2] \\
&= \frac{M-P-\lfloor M\cdot\theta\rceil}{P\,(M-P)}\cdot\frac{\lfloor M\cdot\theta\rceil}{M}\cdot P \\
&\quad + \frac{1}{P\,(M-P)}\cdot\big(\mathbf{Var}[\mathrm{TP}_\theta]+\mathbf{E}[\mathrm{TP}_\theta]^2\big)) \\
&= \frac{(M-P-\lfloor M\cdot\theta\rceil)\cdot\lfloor M\cdot\theta\rceil}{M\,(M-P)} \\
&\quad + \frac{\frac{\lfloor M\cdot\theta\rceil(M-\lfloor M\cdot\theta\rceil)P(M-P)}{M^2(M-1)} + \left(\frac{\lfloor M\cdot\theta\rceil}{M}\cdot P\right)^2}{P\,(M-P)} \\
&= \frac{\lfloor M\cdot\theta\rceil\cdot(M-\lfloor M\cdot\theta\rceil)}{M(M-1)} = \theta^*\cdot(1-\theta^*)\cdot\frac{M}{M-1}
\end{aligned}$$

Of course, since the distribution of TP_θ is known, the expectation of $\mathrm{G}_\theta^{(2)}$ can always be numerically calculated.

Optimal Baselines

Since the function $\varphi:\mathbb{R}\to\mathbb{R}_{\geq 0}$ given by $\varphi(x)=x^2$ is a convex function, we have by Jensen's inequality that

$$\mathbf{E}[\mathrm{G}_\theta^{(2)}]^2 \leq \mathbf{E}\left[\left(\mathrm{G}_\theta^{(2)}\right)^2\right] = \theta^*\,(1-\theta^*)\,\frac{M}{M-1}.$$

This means that

$$\mathbf{E}[\mathrm{G}_\theta^{(2)}] \leq \sqrt{\theta^*\,(1-\theta^*)\,\frac{M}{M-1}}.$$

Therefore, whenever $\theta^*\in\{0,1\}$, then $\mathbf{E}[\mathrm{G}_\theta^{(2)}]\leq 0$. Since $\mathrm{G}_\theta^{(2)}\geq 0$, it must hold that $\mathbf{E}[\mathrm{G}_\theta^{(2)}]=0$. Hence, the set $\{0,1\}$ contains minimisers for $\mathbf{E}[\mathrm{G}_\theta^{(2)}]$. The continuous version of this set is the interval $[0,1/(2M))\cup[1-1/(2M),1]$. To show that this interval contains the only possible values for the minimisers, consider the definition for the expectation of $\mathrm{G}_\theta^{(2)}$:

$$\mathbf{E}\left[\mathrm{G}_\theta^{(2)}\right] = \sum_{k\in\mathcal{D}(\mathrm{TP}_\theta)}\sqrt{\frac{k\cdot((M-P)-(\lfloor M\cdot\theta\rceil-k))}{P\,(M-P)}}\cdot\mathbb{P}(\mathrm{TP}_\theta=k),$$

where $\mathcal{D}(\mathrm{TP}_\theta)$ is the domain of TP_θ, i.e., the set of values k such that $\mathbb{P}(\mathrm{TP}_\theta = k)>0$. Now, let θ be such that $1/(2M)\leq\theta<1-1/(2M)$. Furthermore, consider the summand $S_k^{(\theta)}$ corresponding to $k=\min\{P,\lfloor M\cdot\theta\rceil\}\in\mathcal{D}(\mathrm{TP}_\theta)$:

$$S^{(\theta)}_{k=\min\{P,\lfloor M\cdot\theta\rceil\}} = \begin{cases}\sqrt{\frac{M-\lfloor M\cdot\theta\rceil}{M-P}}\cdot\mathbb{P}(\mathrm{TP}_\theta=P) & \text{if } P\leq\lfloor M\cdot\theta\rceil\\ \sqrt{\frac{\lfloor M\cdot\theta\rceil}{P}}\cdot\mathbb{P}(\mathrm{TP}_\theta=\lfloor M\cdot\theta\rceil) & \text{if } P>\lfloor M\cdot\theta\rceil,\end{cases}$$

which is strictly positive in both cases. Hence, there is at least one term in the summation in the definition of $\mathbf{E}\left[\mathrm{G}_\theta^{(2)}\right]$ that is larger than 0, thus the expectation is strictly positive for $1/(2M) \leq \theta < 1 - 1/(2M)$. Consequently, the minimisation values $\theta_{\min} \in [0, 1]$ are

$$\theta_{\min} \in \underset{\theta \in [0,1]}{\arg\min} \left(\mathbf{E}[\mathrm{G}_\theta^{(2)}]\right) = \left[0, \frac{1}{2M}\right) \cup \left[1 - \frac{1}{2M}, 1\right].$$

Following this reasoning, the discrete form $\theta^*_{\min} \in \Theta^*$ is given by

$$\theta^*_{\min} \in \underset{\theta^* \in \Theta^*}{\arg\min} \left\{\mathbf{E}[\mathrm{G}_\theta^{(2)}]\right\} = \{0, 1\}.$$

5.6.22 Prevalence Threshold (PT)

A relatively new performance measure named *Prevalence Threshold* (PT_θ) was introduced by Balayla [5]. We could not find many articles that use this measure, but it is included for completeness. However, this performance measure has an inherent problem that eliminates the possibility to determine all statistics.

Definition and Distribution

The *Prevalence Threshold* PT_θ is commonly defined as

$$\mathrm{PT}_\theta = \frac{\sqrt{\mathrm{TPR}_\theta \cdot \mathrm{FPR}_\theta} - \mathrm{FPR}_\theta}{\mathrm{TPR}_\theta - \mathrm{FPR}_\theta}.$$

By using the definitions of TPR_θ and FPR_θ in terms of TP_θ (see Equations (5.11) and (5.12)), we obtain:

$$\mathrm{PT}_\theta = \frac{\sqrt{P \cdot (M - P) \cdot \mathrm{TP}_\theta \cdot (\lfloor M \cdot \theta \rceil - \mathrm{TP}_\theta)} - P(\lfloor M \cdot \theta \rceil - \mathrm{TP}_\theta)}{M \cdot \mathrm{TP}_\theta - P \cdot \lfloor M \cdot \theta \rceil}. \tag{5.26}$$

It is clear that this performance measure is not a linear function of TP_θ, therefore we cannot easily calculate its expectation. More importantly, there are fundamental problems with PT_θ.

Division by Zero

Equation (5.26) shows that PT_θ is a problematic measure. When is the denominator zero? This happens when $\mathrm{TP}_\theta = (\lfloor M \cdot \theta \rceil / M) \cdot P$. In this case, the fraction is undefined. Also the numerator is zero in that case. The number of true positives TP_θ can attain the value $(\lfloor M \cdot \theta \rceil / M) \cdot P = \theta^* \cdot P$ whenever the latter is also an integer. For example, this always happens for $\theta^* \in \{0, 1\}$. But even when $\theta^* \in \Theta^* \setminus \{0, 1\}$, PT_θ is still only safe to use when M and P are coprime, i.e., when the only positive integer that is a divisor of both of them is 1. Otherwise, there are always values of $\theta^* \in \Theta^* \setminus \{0, 1\}$ that cause

$\theta^* \cdot P$ to be an integer and therefore PT_θ to be undefined when TP_θ attains that value.

One solution would be to say $\mathrm{PT}_\theta := c$, $c \in [0, 1]$, whenever both the numerator and denominator are zero. However, this c is arbitrary and directly influences the optimisation of the expectation. This makes the optimal parameter values dependent on c, which is beyond the scope of this chapter. Thus, no statistics are derived for the Prevalence Threshold PT_θ.

5.6.23 Threat Score (TS) / Critical Success Index (CSI)

The *Threat Score* [85] TS_θ or *Critical Success Index* [96] is a performance measure that is used for evaluation of forecasting binary weather events: it either happens in a specific location or it does not. It was already used in 1884 to evaluate the prediction of tornadoes [96]. The Threat Score is the ratio of successful event forecasts (TP_θ) to the total number of positive predictions ($\mathrm{TP}_\theta + \mathrm{FP}_\theta$) and the number of events that were missed (FN_θ).

Definition and Distribution

The Threat Score is thus defined as

$$\mathrm{TS}_\theta = \frac{\mathrm{TP}_\theta}{\mathrm{TP}_\theta + \mathrm{FP}_\theta + \mathrm{FN}_\theta}.$$

By using Equations (B2) and (B3), this definition can be reformulated as

$$\mathrm{TS}_\theta = \frac{\mathrm{TP}_\theta}{P + \lfloor M \cdot \theta \rceil - \mathrm{TP}_\theta}.$$

Note that TS_θ is well-defined whenever $P > 0$. The definition of TS_θ is not linear in TP_θ, and so there are no $a, b \in \mathbb{R}$ such that we can write the definition as $Z_\theta\,(a, b)$.

Expectation

Because TS_θ is not linear in TP_θ, determining the expectation is less straightforward than for other performance measures. The definition of the expectation is

$$\mathbf{E}[\mathrm{TS}_\theta] = \sum_{k \in \mathcal{D}(\mathrm{TP}_\theta)} \frac{k}{P + \lfloor M \cdot \theta \rceil - k} \cdot \mathbb{P}(\mathrm{TP}_\theta = k).$$

Unfortunately, we cannot explicitly solve this sum, but it can be calculated numerically.

Optimal Baselines

Although no explicit formula can be given for the expectation, we are able to calculate the extreme values of the expectation and the corresponding optimisers.

Minimal Baseline Firstly, we show that $\theta_{\min} \in [0, \frac{1}{2M})$ constitutes a minimum and that there are no θ outside this interval also yielding this minimum. To this end,

$$\mathbf{E}[\mathrm{TS}_{\theta_{\min}}] = \sum_{k \in \mathcal{D}(\mathrm{TS}_{\theta_{\min}})} \frac{k}{P + 0 - k} \cdot \mathbb{P}(\mathrm{TS}_{\theta_{\min}} = k) = 0,$$

because $\mathcal{D}(\mathrm{TS}_{\theta_{\min}}) = \{0\}$. This is the lowest possible value, since TS_θ is a non-negative performance measure, and hence, $\mathbf{E}[\mathrm{TS}_\theta] \geq 0$ for any $\theta \in [0, 1]$. Now, let $\theta' \geq \frac{1}{2M}$, then there exists a $k' > 0$ such that $\mathbb{P}(\mathrm{TP}_{\theta'} = k') > 0$. Consequently, $\mathbf{E}[\mathrm{TS}_{\theta'}] > 0$ and this means the interval $[0, \frac{1}{2M})$ contains the only values that constitute the minimum. In summary,

$$\min_{\theta \in [0,1]} (\mathbf{E}[\mathrm{TS}_\theta]) = 0,$$
$$\theta_{\min} \in \arg\min_{\theta \in [0,1]} (\mathbf{E}[\mathrm{TS}_\theta]) = \left[0, \frac{1}{2M}\right).$$

Since $\theta^*_{\min}$ is the discretisation of $\theta_{\min}$ it corresponds to 0. More precisely:

$$\theta^*_{\min} \in \arg\min_{\theta^* \in \Theta^*} \{\mathbf{E}[\mathrm{TS}_{\theta^*}]\} = \{0\}.$$

Maximal Baseline Secondly, to determine the maximum of $\mathbf{E}[\mathrm{TS}_\theta]$ and the corresponding parameter $\theta_{\max}$, we determine an upper bound for the expectation, show that this value is attained for a specific interval, and that there is no θ outside this interval also yielding this value. To do this, assume that $\lfloor M \cdot \theta \rceil > 0$. This makes sense, because $\lfloor M \cdot \theta \rceil = 0$ implies $\theta < 1/(2M)$ and such a θ would yield the minimum 0. Now,

$$\begin{aligned}
\mathbf{E}[\mathrm{TS}_\theta] &= \sum_{k \in \mathcal{D}(\mathrm{TP}_\theta)} \frac{k}{P + \lfloor M \cdot \theta \rceil - k} \cdot \mathbb{P}(\mathrm{TP}_\theta = k) \\
&\leq \sum_{k \in \mathcal{D}(\mathrm{TP}_\theta)} \frac{k}{P + \lfloor M \cdot \theta \rceil - P} \cdot \mathbb{P}(\mathrm{TP}_\theta = k) \\
&= \frac{1}{\lfloor M \cdot \theta \rceil} \sum_{k \in \mathcal{D}(\mathrm{TP}_\theta)} k \cdot \mathbb{P}(\mathrm{TP}_\theta = k) = \frac{\mathbf{E}[\mathrm{TP}_\theta]}{\lfloor M \cdot \theta \rceil} \overset{(\mathrm{E})}{=} \frac{P}{M}.
\end{aligned}$$

Next, let $\theta_{\max} \in [1 - 1/(2M), 1]$, then

$$\begin{aligned}
\mathbf{E}[\mathrm{TS}_{\theta_{\max}}] &= \sum_{k = M-(M-P)}^{P} \frac{k}{P + M - k} \cdot \mathbb{P}(\mathrm{TP}_{\theta_{\max}} = k) \\
&= \frac{P}{P + M - P} \cdot \mathbb{P}(\mathrm{TP}_{\theta_{\max}} = P) = \frac{P}{M},
\end{aligned}$$

because $\mathbb{P}(\mathrm{TP}_{\theta_{\max}} = P) = 1$. Hence, the upper bound is attained for $\theta_{\max} \in [1 - 1/(2M), 1]$, and thus, $\theta_{\max}$ is a maximiser.

Now, specifically for $P = 1$, we show that the interval of maximisers is actually $[1/(2M), 1]$. Thus, let $\theta \in [1/(2M), 1 - 1/(2M))$, then $0 < \lfloor M \cdot \theta \rceil < M$ and

$$\begin{aligned}
\mathbf{E}[\mathrm{TS}_\theta] &= \sum_{k=\max\{0,\lfloor M\cdot\theta\rceil-(M-1)\}}^{\min\{1,\lfloor M\cdot\theta\rceil\}} \frac{k}{1+\lfloor M\cdot\theta\rceil - k} \cdot \mathbb{P}(\mathrm{TP}_\theta = k) \\
&= \frac{0}{1+\lfloor M\cdot\theta\rceil - 0} \cdot \mathbb{P}(\mathrm{TP}_\theta = 0) + \frac{1}{1+\lfloor M\cdot\theta\rceil - 1} \cdot \mathbb{P}(\mathrm{TP}_\theta = 1) \\
&= \frac{1}{\lfloor M\cdot\theta\rceil} \cdot \mathbb{P}(\mathrm{TP}_\theta = 1) = \frac{1}{\lfloor M\cdot\theta\rceil} \cdot \left(\frac{\binom{1}{1}\binom{M-1}{\lfloor M\cdot\theta\rceil-1}}{\binom{M}{\lfloor M\cdot\theta\rceil}} \right) = \frac{1}{M},
\end{aligned}$$

which is exactly the upper bound $\mathbf{E}[\mathrm{TS}_{\theta_{\max}}] = P/M$ for $P = 1$.

Next, to show that the maximisers are only in $[1 - 1/(2M), 1]$ for $P > 1$, assume there is a $\theta' < 1 - \frac{1}{2M}$ that also yields the maximum. Hence, there is a $k' \in \mathcal{D}(\mathrm{TP}_{\theta'})$ with $0 < k' < P$ such that $\mathbb{P}(\mathrm{TP}_{\theta'} = k')$. This means

$$\begin{aligned}
\mathbf{E}[\mathrm{TS}_{\theta'}] &= \sum_{k\in\mathcal{D}(\mathrm{TP}_{\theta'})} \frac{k}{P+\lfloor M\cdot\theta'\rceil - k} \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k) \\
&= \frac{k'}{P+\lfloor M\cdot\theta'\rceil - k'} \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k') \\
&\quad + \sum_{k\in\mathcal{D}(\mathrm{TP}_{\theta'})\setminus\{k'\}} \frac{k}{P+\lfloor M\cdot\theta'\rceil - k} \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k) \\
&\leq \frac{k'}{P+\lfloor M\cdot\theta'\rceil - (P-1)} \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k') \\
&\quad + \sum_{k\in\mathcal{D}(\mathrm{TP}_{\theta'})\setminus\{k'\}} \frac{k}{P+\lfloor M\cdot\theta'\rceil - P} \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k) \\
&= \frac{k'}{\lfloor M\cdot\theta'\rceil + 1}\mathbb{P}(\mathrm{TP}_{\theta'} = k') + \sum_{k\in\mathcal{D}(\mathrm{TP}_{\theta'})\setminus\{k'\}} \frac{k}{\lfloor M\cdot\theta'\rceil}\mathbb{P}(\mathrm{TP}_{\theta'} = k) \\
&< \frac{k'}{\lfloor M\cdot\theta'\rceil} \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k') + \sum_{k\in\mathcal{D}(\mathrm{TP}_{\theta'})\setminus\{k'\}} \frac{k}{\lfloor M\cdot\theta'\rceil} \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k) \\
&= \frac{1}{\lfloor M\cdot\theta'\rceil} \sum_{k\in\mathcal{D}(\mathrm{TP}_{\theta'})} k \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k) = \frac{P}{M}.
\end{aligned}$$

Hence, there is a strict inequality $\mathbf{E}[\mathrm{TS}_{\theta'}] < \frac{P}{M}$ and this means θ' is not a maximiser of the expectation. Consequently, the maximisers are only in the interval $[1-1/(2M),1]$ for $P>1$. In summary,

$$\max_{\theta\in[0,1]}\left(\mathbf{E}[\mathrm{TS}_\theta]\right)=\frac{P}{M},$$

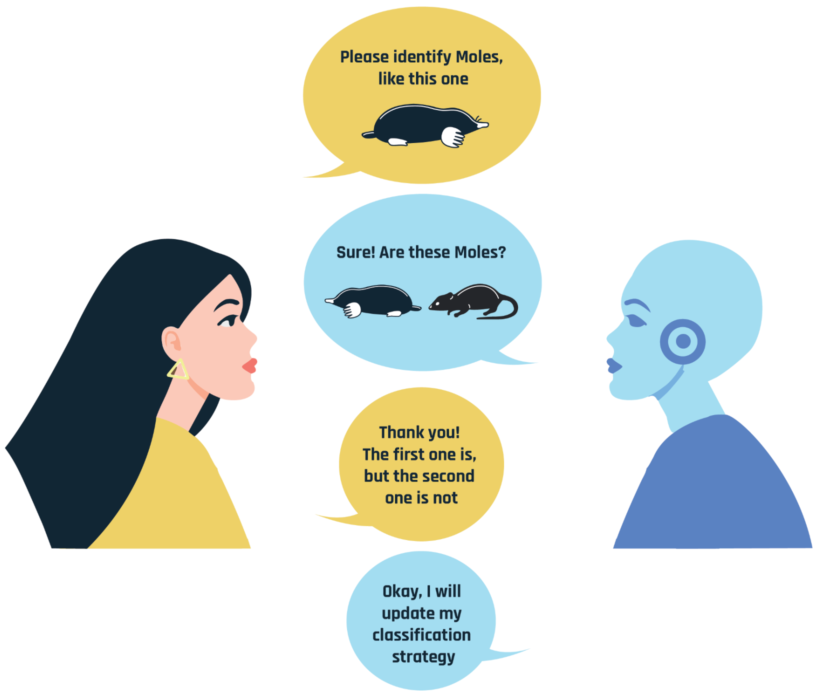
$$\theta_{\max}\in\arg\max_{\theta\in[0,1]}\left(\mathbf{E}[\mathrm{TS}_\theta]\right)=\begin{cases}\left[\frac{1}{2M},1\right] & \text{if } P=1\\ \left[1-\frac{1}{2M},1\right] & \text{if } P>1.\end{cases}$$

Since $\theta^*_{\max}$ is the discretisation of $\theta_{\max}$, we obtain:

$$\theta^*_{\max}\in\arg\max_{\theta^*\in\Theta^*}\left\{\mathbf{E}[\mathrm{TS}_{\theta^*}]\right\}=\begin{cases}\Theta^*\setminus\{0\} & \text{if } P=1\\ \{1\} & \text{if } P>1.\end{cases}$$

Part III

Active Learning


Please identify Moles, like this one
Sure! Are these Moles?
Thank you! The first one is, but the second one is not
Okay, I will update my classification strategy

6 Jasmine: a New Active Learning Approach to Combat Cyber Crime

Luister, je kan echt serieus, je kan ook gewoon één keer naar mij luisteren!

Aaf Brandt Corstius
Wie is de Mol? 2014

One of the reasons that the deployment of network intrusion detection methods falls short is the lack of realistic labelled datasets, which makes it challenging to develop and compare techniques. It is caused by the large amounts of effort that it takes for a cyber expert to classify network connections. This has raised the need for methods that learn from both labelled and unlabelled data which observations are best to present to the human expert. Hence, Active Learning (AL) methods are of interest.

In this chapter, we propose a new hybrid AL method called Jasmine. Firstly, it uses the uncertainty score and anomaly score to determine how suitable each observation is for querying, i.e., how likely it is to enhance classification. Secondly, Jasmine introduces dynamic updating. This allows the model to adjust the balance between querying uncertain, anomalous, and randomly selected observations. To this end, Jasmine is able to learn the best query strategy during the labelling process. This is in contrast to other AL methods in cyber security that all have static, predetermined query functions. We show that dynamic updating, and therefore Jasmine, is able to consistently obtain good and more robust results than querying only uncertainties, only anomalies, or a fixed combination of the two.

Based on [45]:
Jan Klein, Sandjai Bhulai, Mark Hoogendoorn, and Rob van der Mei
Jasmine: a New Active Learning Approach to Combat Cybercrime
2022 Machine Learning with Applications 9

6.1 Introduction

The fight against cyber crime has become a priority for many countries. This is with good reason, because the average cost of a single cyber attack in Europe is around 50 thousand euros, as estimated by Forrester Consulting and Hiscox [19]. Most common attacks were mostly targeted on companies and even governmental institutes. For example, 68% of Dutch firms reported at least one cyber incident in 2019. Therefore, the amount of money that the surveyed European companies invested in cyber security has increased by 39% from 1.3 million to 1.8 million euros. In academia, the awareness for research in the field of cyber security has also grown. For instance, Mouloua et al. [77] analysed the articles published in the Proceedings of the Human Factors and Ergonomics Society (HFES) from 1980 to 2018. They showed that 73% of articles related to cyber security published in that almost 40-year span were written in the last nine years.

Since most cyber attacks are aimed at companies and countries, it is important to know how networks of computers can be protected. A Network Intrusion Detection System (NIDS) is software designed to detect unusual, malicious events in a computer network. There are several types of systems with each having their own set of challenges for which several solutions have been proposed [103, 107, 131]. However, there are some broader challenges in cyber security research. Most importantly, Xin et al. [124] and Yang et al. [125] argue that not much consideration is given to deployment efficiency. This means that little is practically done with published research, because of time complexity of the techniques and the efficiency of detection in actual networks. The latter is due to the lack of realistic datasets, since it takes a lot of time and effort for a human to classify network connections correctly. Moreover, cyber analysts have to label many redundant connections just to construct a representative dataset. This loads the experts with tedious work and leads to an insufficient use of their capabilities. Therefore, it would be beneficial to only present 'informative' network connections to the cyber analyst. This can be realised in Active Learning (AL), in which the model chooses from which unlabelled data instances it wants to learn and then queries their labels [52, 100].

Several AL methods have been proposed in network intrusion detection (NID) research. Many of them focus on querying *uncertain* data, i.e., requesting the label of observations about which the model is not sure how to classify them [36, 39, 60]. Adding these observations with their correct label is expected to enhance classification performance more quickly than randomly selecting observations. Other studies consider different query strategies or combine several query approaches to make the AL procedure more robust [128]. Stokes et al. [104] query both uncertain and *anomalous* instances. The latter are observations that behave vastly differently than expected and that could indicate malicious activities. Although combining query types increases prediction performance, the optimal balance of the types depends on, for example, the dataset. Moreover, the balance has to be determined beforehand and is still up for debate.

In our research, we propose a novel AL method called Jasmine that introduces *α-dynamic updating*. This allows our method to adjust the balance between querying different types of observations such that the right types are proposed to the human expert at the right time. This makes sense, because the structure of the labelled set (on which the classifier is trained) changes during the labelling procedure. Hence, Jasmine is able to learn the best query strategy during the process. The types of instances that Jasmine considers potentially informative are uncertain, anomalous and randomly selected observations. Jasmine is able to dynamically change the balance between the three types to ensure that the most informative observations to the NID model are queried. This sets our method apart from existing AL methods, because (to our knowledge) Jasmine is the only method that allows for this. Our contributions are:

- We propose a new AL method called Jasmine that introduces dynamic updating of the balance between querying uncertain, anomalous, and randomly selected unlabelled data.
- We present the mathematical formulation of Jasmine and explain why it can find a good balance given the current labelling state.
- We apply Jasmine to two commonly used NID datasets and use them in different experimental settings. We show that Jasmine obtains good results and is more robust than static query approaches. Moreover, it performs even better in the case of highly imbalanced data. Therefore, Jasmine is more reliable to use, because it can adapt itself to different situations.

The rest of this chapter is organised as follows. In Section 6.2, we explain the AL paradigm and what methods have been proposed in NID. This has some overlap with Section 1.3 and parts could therefore be skipped. Section 6.3 introduces the mathematical notation used in the AL framework. In Section 6.4, we propose our method Jasmine and explain how it works. The experiments that we execute to validate our method are discussed in Section 6.5. Section 6.6 subsequently shows the results of these experiments and their interpretations. Finally, in Section 6.7, we draw conclusions about this study and make suggestions for further research.

6.2 Related Work

Active Learning is a subfield of Machine Learning (ML) in which the premise is that only a small part of the data is labelled, while the labels for the rest of the data are not specified. The AL procedure is illustrated in Figure 6.1. Firstly, an ML model is trained on the labelled set and applied to the unlabelled observations to obtain output predictions. Then, interesting instances from the unlabelled part are selected and an oracle is asked to provide the actual

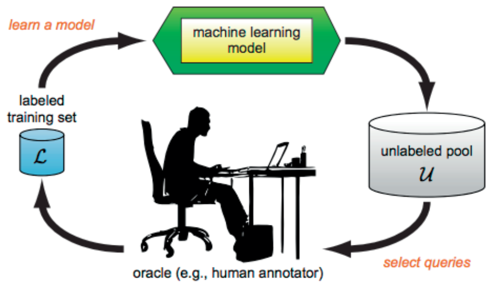


Figure 6.1: Illustration of Active Learning framework [100]

labels. Subsequently, these query observations are added to the labelled set and the procedure continues. An important advantage of AL is that the model needs less training data to learn better [52, 100]. For instance, it has been used to improve performance in the diagnosis and treatment of diseases [9], in the personalisation and hybridisation of recommender systems [25], and in community detection [31]. Moreover, AL is especially beneficial in domains in which it is laborious to label data. Examples of such fields are speech recognition, information extraction (person names from texts, annotation of genes, etc.), classification of files (relevant or irrelevant documents, images, etc.) [100], and NID. The diversity among these examples shows the broad application capability of AL.

6.2.1 Query Strategies

There are multiple ways to query the oracle, but we focus on *pool-based sampling*. Here, the decision to query certain observations is based on some informativeness measure that is calculated for all instances in the unlabelled pool (or a subset thereof). This approach to querying has been applied to many real-world problems, including many of the application domains mentioned before. It works well with a human oracle and when it is relatively easy to compute the informativeness of all observations at once. A commonly used measure for this is the uncertainty score. This score is used to query observations about which the model is the least certain on how to predict their label [58]. This is a simple informativeness measure, because no new models have to be trained and only the output of the classifier is required to determine the uncertainty of each observation.

6.2.2 Active Learning in Network Intrusion Detection

Which specific query strategy and ML technique are used depends on the AL application. In NID, several AL methods have been proposed. These approaches mostly rely on uncertainty sampling. To illustrate this, Li and Guo [60] use Transductive Confidence Machines for K-Nearest Neighbors for supervised NID with uncertainty sampling and query by committee; Görnitz et al. [36] use a Support Vector Domain Description for Anomaly Detection (AD) with uncertainty sampling as the AL component; and Guerra Torres et al. [39] make use of Random Forests for prediction and query uncertain observations. These studies show that a method with AL obtains better results than one without it, or that the proposed query strategy performs better than randomly presenting observations to the oracle. Though, the diversity in considered query approaches is low since uncertainty sampling is often chosen. Both Gu and Zydek [38] and Yang et al. [125] present overviews of multiple other query strategies for NID, but the authors ultimately choose for uncertainty sampling too. Yet, literature does show that this common query approach performs well.

However, there is room for significant improvement. For example, what happens when more than one informativeness measure is considered for query selection? There are studies that incorporate two measures to improve classification performance. Stokes et al. [104] propose to combine uncertainty sampling with querying anomalous data, thus presenting instances that behave vastly differently than expected to the oracle. Yin et al. [128] combine the uncertainty informativeness measure with density information as the query approach. More specifically, the distance of an observation to its nearest neighbour is calculated and instances residing in high density areas are more likely to be queried. These two studies show that using multiple informativeness measures further improves performance, because more characteristics of the data are used to determine which unlabelled observations would improve predictions the most if they were labelled.

The common factor in the aforementioned research is that all proposed query functions are *static* in nature. From the start, it is exactly known how the set of query observations is constructed and this approach cannot be changed. This means the contribution of each informativeness measure in the selection of the query observations has to be fixed beforehand. However, the optimal balance of these contributions depends on the overall structure of the data and also on the current state of the labelling process. Therefore, we consider a *dynamic* query approach to address these problems. To this end, the balance can be adapted during the procedure to best fit the data. More specifically, the method learns the distribution of query types that is expected to increase prediction performance the most given the current state.

6.2.3 ALADIN

The AL methodology for NID that we use as a starting point was developed by Stokes et al. [104] and is called ALADIN. It was chosen because it combines two important informativeness measures: the *(i)* uncertainty score and *(ii)* anomaly score. On the one hand, by querying selected anomalies, new classes of network traffic can be found within the data. On the other hand, by querying uncertainties, the accuracy of the classifier should increase in the next time step when the correct classes have been provided by the human expert. ALADIN

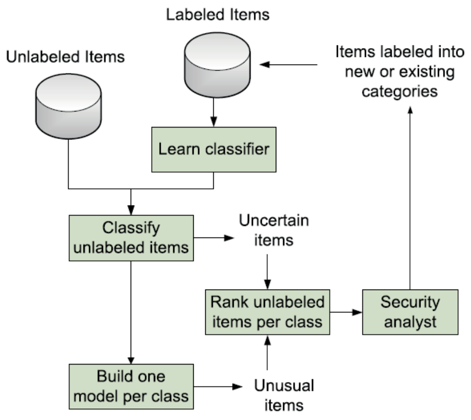


Figure 6.2: Schematic representation of an ALADIN iteration [104]

combines both Pelleg's algorithm for AD [88] and Almgren's algorithm for NID [3]. In the first algorithm, an AD model is constructed for each data class, so one for the benign class and one for the malicious class in the binary case. How unlikely an unlabelled observation is according to the model of its (predicted) class, determines how anomalous it is. The less likely an instance is, the more interesting it is for querying. In the second algorithm, observations that lie close to the decision boundary of the trained classifier are deemed uncertain, and hence, interesting for querying. The description of ALADIN is illustrated in Figure 6.2. On the 1999 KDD-Cup dataset [106], Stokes et al. [104] show that ALADIN achieved high accuracy in predicting known traffic classes. Moreover, it was able to detect previously unknown categories quickly. This shows that incorporating anomalous and uncertain observations in the query set led to favourable results.

ALADIN uses simple ML techniques such as logistic regression and naive Bayes to be able to scale well. However, the authors mention that the benign class in the KDD-Cup dataset is very diverse, meaning that this class may not be easily predicted with the logistic regression classifier. In our research, we consider more advanced ML techniques that are better able to capture the diversity of network traffic. More importantly, in ALADIN, half of the queried observations are anomalous, while the other half is uncertain. This fixed 50/50 split is not motivated by the authors. Hence, model performance can improve when the proportion between querying anomalies and uncertainties is changed depending on the considered dataset and within the process. This leads to our α-dynamic updating.

6.3 Preliminaries

Before we provide the mathematical formulation of Jasmine, we introduce some notation to describe the AL framework in general. Firstly, we assume to have a dataset $\mathbf{X} \in \mathbb{R}^{M \times K}$, with $M \in \mathbb{N}$ the number of observations and $K \in \mathbb{N}$ the number of features, or attributes. Let $\mathbf{x}_i \in \mathbb{R}^K$ be the feature vector of observation $i \in \{1, \ldots, M\}$, and let $y_i \in \{0, 1, *\}$ be its corresponding response value. Here, '0' represents the benign class and '1' the malicious class. The symbol '*' means that the class, or label, is missing. In AL, it is assumed that only a (possibly small) part of the data is labelled, while the rest is unlabelled. The set of labelled observations $\mathcal{L}(t)$ and the set of unlabelled observations $\mathcal{U}(t)$ depend on the iteration or time step $t = 1, \ldots, T$, where $T \in \mathbb{N}$ is a predetermined maximum number of iterations. Since more labels become available when the iteration procedure progresses, the vector of response values of all the instances $\mathbf{y}(t) = (y_1(t), \ldots, y_M(t))$ is dependent on time. Now, for all iterations it holds that $\mathcal{L}(t)$ and $\mathcal{U}(t)$ are disjoint and their union is the complete dataset $(\mathbf{X}, \mathbf{y}(t))$ with labels up to time t. Let $L(t) := |\mathcal{L}(t)|$ be the number of labelled observations and $U(t) := |\mathcal{U}(t)|$ the number of unlabelled instances at time t. Note that $L(t{+}1) > L(t)$, while $U(t{+}1) < U(t)$, because every iteration the human expert adds labels to previously unlabelled observations. $\mathcal{L}(0)$ contains the instances that are labelled from the start, while $\mathcal{U}(0)$ consists of all initially unlabelled observations. In iteration t, a supervised ML technique $f_t : \mathbb{R}^K \to [0, 1]$ is trained on $\mathcal{L}(t-1)$. This classifier f_t is then applied to $\mathcal{U}(t-1)$ to obtain predictions for the actual classes of the unlabelled observations. Then, a query function ψ determines which instances from $\mathcal{U}(t-1)$ are selected to be queried to the human expert. Let $\mathcal{Q}(t) \subseteq \mathcal{U}(t-1)$ be the constructed set of query observations. Usually, this set has fixed size $Q := |\mathcal{Q}(t)|$. Then, the expert provides the labels $y_q(t) \in \{0, 1\}$ for the observations $q \in \mathcal{Q}(t)$. Note that in the previous iteration their labels were still unknown: $y_q(t-1) =$ '*'.

6.4 Methodology

In this section, we introduce our AL method Jasmine. Its key component is α-dynamic updating, which allows the model to dynamically adjust the balance between querying anomalous, uncertain, and random observations during the procedure. The adjustment of the balance comes in two flavours. Firstly, the initial proportions of the three types of query observations are determined. Based on the available initially labelled data $\mathcal{L}(0)$, it can be beneficial to start with querying 60% anomalous, 15% uncertain, and 25% random observations, for example. Secondly, the balance between the types can be changed during the labelling process, because it may be better to query increasingly more anomalous instances when more and more labels are provided by the oracle. This leads to dynamically updating the query fractions. Moreover, we also consider querying random observations. This may seem counter-intuitive in the AL setting, because its premise is not to bother the human expert with labelling redundant observations. However, when $\mathcal{L}(0)$ is not a good representation of the entire dataset, querying some random observations could be beneficial.

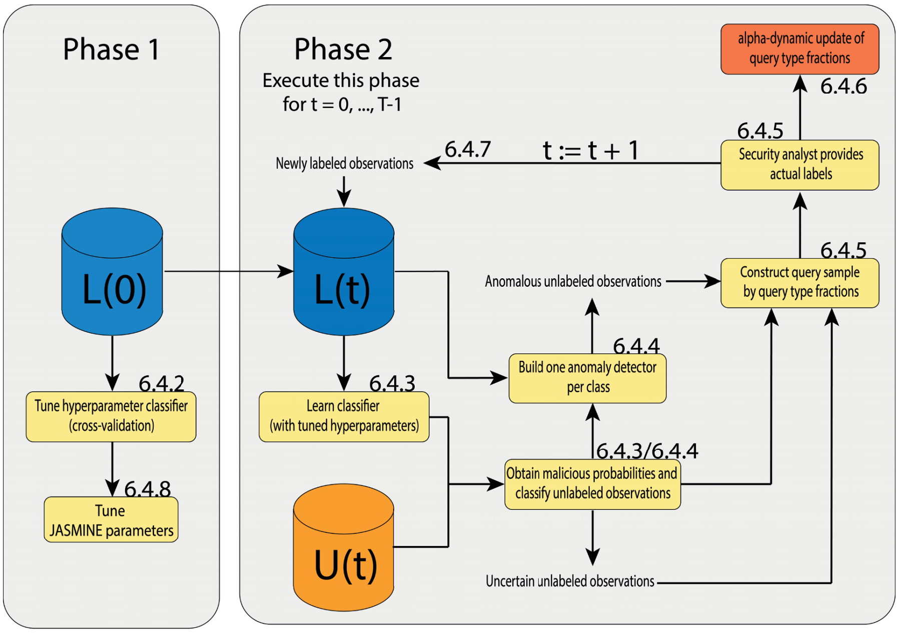

Phase 1
Phase 2
Execute this phase for t = 0, ..., T-1
alpha-dynamic update of query type fractions
6.4.6
6.4.5
Security analyst provides actual labels
6.4.7
t := t + 1
Newly labeled observations
L(0)
L(t)
6.4.5
Construct query sample by query type fractions
Anomalous unlabeled observations
6.4.4
Build one anomaly detector per class
6.4.2
Tune hyperparameter classifier (cross-validation)
6.4.3
Learn classifier (with tuned hyperparameters)
6.4.3/6.4.4
Obtain malicious probabilities and classify unlabeled observations
6.4.8
Tune JASMINE parameters
U(t)
Uncertain unlabeled observations

Figure 6.3: Schematic representation of complete Jasmine procedure. The numbers of the form ‘6.4.x’ denote in which subsection the corresponding component is explained

The complete Jasmine procedure is illustrated in Figure 6.3. It consists of two consecutive phases. In Phase 2, the actual AL component of Jasmine is executed given the results of Phase 1. More specifically, the classifier is trained on $\mathcal{L}(t)$ and applied to $\mathcal{U}(t)$ to obtain malicious probabilities (Section 6.4.3). Then, the informativeness measures of the unlabelled observations are calculated (Section 6.4.4) and the query sample is constructed based on these measures (Section 6.4.5). Next, the actual labels are provided by the human oracle. Then, the crucial part of Jasmine is performed by updating the query fractions based on the results obtained during the iteration (Section 6.4.6). Consequently, the query set of the next iteration should have a better balance of anomalous, uncertain, and random observations. Finally, the labelled query observations are added to $\mathcal{L}(t+1)$ (Section 6.4.7) and the procedure continues. In Phase 1, good values of the hyperparameters of the classifier are determined (Section 6.4.2). Also, good values of the Jasmine-specific parameters are chosen in this phase (Section 6.4.8). Since tuning of the Jasmine parameters is a reduced version of the second phase, we explain the latter first in this section.

6.4.1 Classification and Anomaly Detection Techniques

The supervised ML technique used as the classifier in Jasmine is the Gradient Boosting Machine (GBM), which was designed by Friedman [30]. One of the main merits of the GBM is its flexibility and robustness with respect to the number of hyperparameters [11, 82, 84]. It can easily be customised for different practical purposes. Another advantage of the GBM is its interpretability. Since the technique is an ensemble of multiple simple models, it can be easily understood.

The AD method used in Jasmine is the Isolation Forest (IF), which was developed by Liu et al. [65]. Just as the GBM, it is a tree-based ensemble ML technique. Most AD models try to construct a representation of normal behaviour, but the IF aims to isolate the anomalous observations. The authors use two important properties of anomalies: *(i)* there are few of them, and *(ii)* they have distinctly different characteristics than the majority of observations. Because these properties hold, it should be relatively simple to isolate them from the rest. The main advantage of the IF technique is that it is fast and easy to interpret [64].

6.4.2 Tuning GBM Hyperparameters

The GBM has hyperparameters that are determined beforehand. In Jasmine, good values are found via hyperparameter tuning [18]. This is done on the set $\mathcal{L}(0)$, because by definition the observations in this set are the only ones with labels from the start. We use k-fold cross validation such that each observation is both utilised for training and evaluating the model. This is desirable, because $\mathcal{L}(0)$ is usually rather small, so we want to effectively use each provided label. During tuning, also the computation time is taken into account, because the GBM is retrained each AL iteration and training times quickly stack. We want to find hyperparameters that yield good performance in both predictive ability and computation time. To this end, assume there are H hyperparameter combinations yielding a sequence of performance metrics $h_1, \dots, h_H$ and a sequence of computation times $t_1, \dots, t_H$. Let $h_* := \max\{h_1, \dots, h_H\}$ be the best performance, and let $j_* := \arg\max\{h_1, \dots, h_H\}$ be the combination yielding this performance. If there are several combinations resulting in the best performance metric, then j_* is the one with the smallest computation time t_{j_*}. Also, if there is a combination yielding a performance of almost h_*, but with a much smaller computation time than t_{j_*}, then we prefer to go for that combination. In that case, combination j is chosen as the optimal one whenever

$$d_j := \frac{h_{j^*} - h_j}{t_{j^*} - t_j} < \varepsilon,$$

where $\varepsilon > 0$ is some predefined threshold. If several j satisfy this requirement, then the one with the smallest d_j is chosen.

6.4.3 Training, Evaluating, and Predicting

The classification is done by means of a GBM, which we described in Section 6.4.1. The hyperparameter values for this technique are determined by tuning, as described in Section 6.4.2. In iteration t, the GBM is trained with k-fold cross validation on $\mathcal{L}(t-1)$, yielding the classifier f_t and some threshold probability $\theta \in (0,1)$. This θ represents the border between predicting an instance as 0 (benign) or as 1 (malicious), and is the value that maximises some performance metric on $\mathcal{L}(t-1)$. In Jasmine, we are interested in how confident f_t is in its predictions. The closer the predicted probability is to θ, the less certain the model is. Intuitively, a value of 0.5 can be seen as the least certain probability for a binary classifier, because it is precisely between 0 and 1. Therefore, θ is transformed to be 0.5 and the probabilities are changed accordingly. We provide more reasoning why this is done in Section 6.4.4. The function $\varphi_\theta : [0,1] \to [0,1]$, defined as

$$\varphi_\theta(y) = \frac{(1-\theta)y}{(1-2\theta)y+\theta},$$

is applied to all predicted probabilities to transform them. We chose this function, because it has the following necessary and desirable properties: *(i)* φ is continuously differentiable (smooth), *(ii)* $\varphi_\theta(0) = 0$, *(iii)* $\varphi_\theta(\theta) = 0.5$, *(iv)* $\varphi_\theta(1) = 1$, and *(v)* $\varphi'_\theta(y) \geq 0$. Note that the probabilities do not change when θ is already 0.5: $\varphi_{\theta=0.5}(y) = y$ for all $y \in [0,1]$.

After this, f_t is applied to the unlabelled set $\mathcal{U}(t-1)$ to obtain predicted probabilities $\hat{y}_u(t) \in [0,1]$ (which have been transformed by φ_θ) for each $u \in \mathcal{U}(t-1)$. Consequently, the predicted class $\hat{c}_u(t)$ is 0 if $\hat{y}_u(t) < 0.5$ and 1 if $\hat{y}_u(t) \geq 0.5$.

6.4.4 Calculating Certainty Score and Anomaly Score

In the next step, the measures needed to determine which unlabelled observations to present to the expert are calculated. These measures are the certainty score and anomaly score.

The certainty score $z_u(t) \in [0,1]$ is defined as

$$z_u(t) := 2\left|\hat{y}_u(t) - \frac{1}{2}\right|, \tag{6.1}$$

for each $u \in \mathcal{U}(t-1)$. The lower this score, the more uncertain the trained model f_t is about the predicted label of u. Equation (6.1) is the commonly used definition for the certainty score in AL methods, such as in ALADIN. It also shows why we transformed the raw predicted probabilities by φ_θ. Now, the distance from $\frac{1}{2}$ can be at most 0.5 in both the benign direction (corresponding to $\hat{y}_u(t) \downarrow 0$) and the malicious direction ($\hat{y}_u(t) \uparrow 1$), making $z_u(t)$ a symmetric score.

The IF technique that we described in Section 6.4.1 is used to determine the anomaly score $a_u(t)$ for each $u \in \mathcal{U}(t-1)$. Firstly, the class-specific observation sets are defined for each $c \in \{0, 1\}$ by

$$\begin{aligned} \mathcal{L}^{(c)}(t-1) &:= \{l \in \mathcal{L}(t-1) : y_l = c\}, \\ \mathcal{U}^{(c)}(t) &:= \{u \in \mathcal{U}(t-1) : \hat{c}_u(t) = c\}. \end{aligned} \tag{6.2}$$

Now, one IF is trained on the set $\mathcal{L}^{(0)}(t-1) \cup \mathcal{U}^{(0)}(t)$ and one on the set $\mathcal{L}^{(1)}(t-1) \cup \mathcal{U}^{(1)}(t)$, yielding a benign IF $I_t^{(0)} : \mathbb{R}^K \to [0, 1]$ and a malicious IF $I_t^{(1)} : \mathbb{R}^K \to [0, 1]$, respectively. Then, the anomaly score of $u \in \mathcal{U}(t-1)$ is defined as

$$a_u(t) := I_t^{(\hat{c}_u(t))}(\mathbf{x}_u). \tag{6.3}$$

This is the output value that the appropriate IF produces when the feature vector $\mathbf{x}_u$ is fed into it.

6.4.5 Constructing Query Sample $\mathcal{Q}(t)$

An important component of Jasmine is the way in which the query function ψ^{Jas} determines how the query sample $\mathcal{Q}(t)$ is constructed. There are three types of query observations: anomalies, uncertainties, and random instances. Which part of $\mathcal{Q}(t)$ should be allocated to which type is given by the anomaly fraction $\alpha_a(t)$, uncertainty fraction $\alpha_z(t)$, and randomness fraction $\alpha_r(t)$. For each class $c \in \{0, 1\}$, the unlabelled observations in $\mathcal{U}^{(c)}(t)$ (see (6.2)) are sorted from most anomalous to least anomalous. Then, the top $\frac{1}{2}\alpha_a(t) \cdot Q$ (rounded to the nearest integer) observations are taken as the anomalous query instances for predicted class c. The uncertainties are selected similarly: the observations in $\mathcal{U}^{(c)}(t)$ are sorted from least certain to most certain for each class c. Then, the top $\frac{1}{2}\alpha_z(t) \cdot Q$ instances are selected for the uncertain query observations for predicted class c. The random query observations are selected in a simple way: a sample of size $\alpha_r(t) \cdot Q$ is taken from $\mathcal{U}(t)$ (without the already selected anomalous and uncertain observations).

There are some important technicalities in constructing $\mathcal{Q}(t)$. First of all, it is possible that there are not enough observations for a specific class. Then, observations from the other class are added to reach the number of required anomalies or uncertainties. Secondly, there can be an overlap between the most anomalous and least certain instances: some observations can be both anomalous and uncertain, and hence, we select them for both query types.

Ultimately, when the query observations have been determined, the query set $\mathcal{Q}(t)$ is shown to the human expert and they provide the correct label y_q for each $q \in \mathcal{Q}(t)$.

6.4.6 α-dynamic Update

Constructing Update Parameters

The query set $\mathcal{Q}(t)$ obtained in Section 6.4.5 can be decomposed in $\mathcal{Q}_a(t)$, $\mathcal{Q}_z(t)$, and $\mathcal{Q}_r(t)$, which are the sets of anomalous, uncertain, and random queries, respectively. Also, let $Q_a(t)$, $Q_z(t)$, and $Q_r(t)$ be their respective sizes. Furthermore, let $y_q \in \{0,1\}$ be the real label of query observation q, $\hat{y}_q(t) \in [0,1]$ its predicted malicious probability, and $\hat{c}_q(t) \in \{0,1\}$ its predicted label. In practice, the real label is available since the cyber expert provides it. We want to construct a metric for the anomalous queries and one for the uncertain queries that describe how much information the observations of those types can potentially add to the model. We do this by looking at the false negatives (FNs) and false positives (FPs) in $\mathcal{Q}(t)$. The reasoning is that when there are many FNs and FPs, then the model was bad at predicting the real classes of those unlabelled observations. Hence, adding these observations to the labelled set $\mathcal{L}(t)$ could yield a lot of information for the classifier trained in the next iteration. In short, we consider the fraction of FPs and FNs in $\mathcal{Q}(t)$, but also take a look at how convincing they are. If the model is fairly certain that an observation is malicious while it is in fact benign, then this FP obtains a larger weight than if the model is not that sure. Consequently, we define the anomaly information metric $\delta_a^\beta(t)$ as follows:

$$\delta_a^\beta(t) := \frac{\sum_{q\in\mathcal{Q}_a(t)} |\hat{y}_q(t) - y_q| \cdot \left(\beta \cdot \mathbf{1}_{\{\hat{c}_q=0, y_q=1\}} + \mathbf{1}_{\{\hat{c}_q=1, y_q=0\}}\right)}{Q_a(t) + (\beta - 1) \cdot |\{q \in \mathcal{Q}_a(t) : y_q = 1\}|}. \tag{6.4}$$

Note that the first indicator function in (6.4) corresponds to an FN: the real label is 1, but the predicted label is 0. Equivalently, the second indicator function corresponds to an FP. If the query observation q is an FN, it gets weighted by some parameter $\beta > 0$. This enables us to put more ($\beta > 1$) or less ($\beta < 1$) emphasis on the FNs compared to the FPs. Thus, if q is an FN, then it obtains value $\beta|\hat{y}_q(t) - y_q|$; if it is an FP, it obtains $|\hat{y}_q(t) - y_q|$. The farther the predicted probability $\hat{y}_q(t)$ is from 0 or 1 (thus the less certain the model is about the prediction), the larger the assigned value for q becomes. The denominator ensures that the value of $\delta_a^\beta(t)$ is at most 1. When there are no FPs and FNs, then the value of the metric is 0. Consequently, $\delta_a^\beta(t) \in [0,1]$. The larger $\delta_a^\beta(t)$ is, the more information the anomalies convey, because we expect that incorrectly predicted observations add relevant information to the model if they are added to the train set with the correct labels.

Similarly, we define the uncertainty information metric

$$\delta_z^\beta(t) := \frac{\sum_{q\in\mathcal{Q}_z(t)} |\hat{y}_q(t) - y_q| \cdot \left(\beta \cdot \mathbf{1}_{\{\hat{c}_q=0, y_q=1\}} + \mathbf{1}_{\{\hat{c}_q=1, y_q=0\}}\right)}{Q_z(t) + (\beta - 1) \cdot |\{q \in \mathcal{Q}_z(t) : y_q = 1\}|}, \tag{6.5}$$

as a measure on how much information the uncertain observations in the query set add on average. Also, $\delta_z^\beta(t) \in [0,1]$.

Next, the difference between the information metrics that we defined in (6.4) and (6.5) describes whether anomalies or uncertainties could add more information to the model. We define this difference $\Delta(t)$ as

$$\Delta(t) := \delta_a^\beta(t) - \delta_z^\beta(t) \in [-1, 1].$$

It is used to determine how the query fractions of anomalies and uncertainties are updated. If $\Delta(t) > 0$, then the queried anomalies could add more information on average, and hence, preferably, more anomalies are selected the next iteration. If $\Delta(t) < 0$, then the uncertainties could add more information, and so, more uncertainties are selected.

Now, we want the possibility to put more or less emphasis on bigger or smaller values of $\Delta(t)$. Hence, we introduce a non-linearity governed by $\gamma > 0$ to obtain $\Delta^\gamma(t)$. To this end, we define the update factor as

$$\Delta^\gamma(t) := \operatorname{sgn}(\Delta(t)) \cdot |\Delta(t)|^{1/\gamma}. \tag{6.6}$$

When $\gamma = 1$, then (6.6) reduces to the linear case $\Delta^\gamma(t) = \Delta(t)$. If $0 < \gamma < 1$, then $|\Delta^\gamma(t)| \leq |\Delta(t)|$, so the update factor is relatively smaller. On the other hand, if $\gamma > 1$, then $|\Delta^\gamma(t)| \geq |\Delta(t)|$ and the factor is relatively larger.

Defining Query Fractions

The update factor $\Delta^\gamma(t)$ is used to determine whether more anomalies or uncertainties should be queried in the next iteration. The updates $\alpha_a(t+1)$ and $\alpha_z(t+1)$ have the forms

$$\alpha_a(t+1) = \lambda_{t+1} \left(\alpha_a(t) + w_a^{(1)}(t) \cdot \max\{0, \Delta^\gamma(t)\} + w_a^{(2)}(t) \cdot \min\{0, \Delta^\gamma(t)\} \right), \tag{6.7}$$

and

$$\alpha_z(t+1) = \lambda_{t+1} \left(\alpha_z(t) + w_z^{(1)}(t) \cdot \max\{0, -\Delta^\gamma(t)\} + w_z^{(2)}(t) \cdot \min\{0, -\Delta^\gamma(t)\} \right), \tag{6.8}$$

respectively. Here, the w variables are non-negative constants that we need to ensure that the fractions stay within the correct bounds for time step t. Moreover, λ_{t+1} denotes a linear function which guarantees that the updated fractions are also within the bounds for time step $t+1$. Let us examine (6.7) in more detail: the new fraction $\alpha_a(t+1)$ is based on the old value $\alpha_a(t)$ plus some value when $\Delta^\gamma(t) > 0$ or minus some value when $\Delta^\gamma(t) < 0$. In other words, when the anomalies add more information on average than the uncertainties, then $\alpha_a(t+1)$ becomes larger. Otherwise, $\alpha_a(t+1)$ becomes smaller. The update dynamics are the other way around for (6.8). We provide the explicit mathematical definitions of the w variables and λ_{t+1} in Section 6.8. It is important to mention that they depend on the hyperparameters $\alpha_a^{(0)}$ and $\alpha_z^{(0)}$, which are the initial query fractions of anomalies and uncertainties, respectively.

The fraction of random observations that we want to query relates to the number of available labelled observations $L(t)$. If $L(0)$ is small, then $\mathcal{L}(0)$ is less likely to be a good representation of the complete labelled dataset $(\mathbf{X}, \mathbf{y})$. In that case, we want to query relatively more random instances at the start to be able to obtain a representative train set for the GBM classifier to learn from. As $L(t)$ increases, we want to query fewer and fewer random observations and let AL take over in the sense of querying anomalies and uncertainties. Therefore, let $\alpha_r(t)$ be an exponentially decreasing function in t. More specifically,

$$\alpha_r(t) = \alpha_r^{\max} \cdot 2^{-\tau \cdot L(t)}, \tag{6.9}$$

with $\tau > 0$ the decrease speed and $L(t) = L(0) + Q \cdot t$. Note that this function is determined beforehand. Hence, how the fraction of randomly queried observations changes, is fixed during the Jasmine procedure. The value of $\alpha^{\max}$ is directly determined by the hyperparameters $\alpha_a^{(0)}$ and $\alpha_z^{(0)}$.

It is important to see that executing the α-dynamic update step does not take much computation time. No additional ML models have to be trained and only relatively simple operations are performed to determine the values of the query fractions for the next time step.

6.4.7 Final Iteration Updates

The last steps that Jasmine has to perform are rather simple, the labelled query set $\mathcal{Q}(t)$ is added to the current labelled set $\mathcal{L}(t-1)$ and removed from the current unlabelled set $\mathcal{U}(t-1)$. More specifically, $\mathcal{L}(t) := \mathcal{L}(t-1) \cup \mathcal{Q}(t)$ and $\mathcal{U}(t) := \mathcal{U}(t-1) \setminus \mathcal{Q}(t)$. The complete Jasmine procedure is summarised in Algorithm 1.

6.4.8 Tuning Jasmine Hyperparameters

Chronologically speaking, tuning of the Jasmine-specific hyperparameters takes place in Phase 1 before the actual AL procedure in Phase 2, as we illustrated in Figure 6.3. To be more specific, *Jasmine tuning* occurs after tuning the hyperparameters for the GBM, and thus, directly after Section 6.4.2. Recall that the hyperparameters are *(i)* $\alpha_a^{(0)}$ and $\alpha_z^{(0)}$, the starting fractions of querying anomalous and uncertain observations, respectively; *(ii)* β, the parameter assigning a certain weight to an FN compared to an FP, given in (6.4) and (6.5); *(iii)* γ, the update magnitude, given in (6.6); and *(iv)* τ, the decrease speed in querying random observations, given in (6.9). Using normalisation, $\alpha_a^{(0)}$ and τ completely determine $\alpha_z^{(0)}$, so the latter does not have to be tuned.

To determine appropriate values for these hyperparameters, the initially labelled set $\mathcal{L}(0)$ is randomly partitioned into the sets $\mathcal{L}_J(0)$, $\mathcal{U}_J(0)$, and $\mathcal{E}_J$. During Jasmine tuning, $\mathcal{L}_J(0)$ is taken as the initially labelled set, $\mathcal{U}_J(0)$ as the initially unlabelled set, and $\mathcal{E}_J$ as the evaluation set. Let $\mathcal{J}$ be the set with hyperparameter values that we want to consider for tuning. Thus, an

Algorithm 1 Jasmine procedure

```
Require: Labelled set $\mathcal{L}(0)$, unlabelled set $\mathcal{U}(0)$, number of iterations $T$, query function $\psi^{\text{Jas}}$
 1: Tune parameters of GBM on $\mathcal{L}(0)$ (Section 6.4.2)
 2: Tune Jasmine-specific hyperparameters $(\alpha_a^{(0)}, \beta, \gamma, \tau)$ on $\mathcal{L}(0)$ (Section 6.4.8) and determine $\alpha_z^{(0)}$ by normalisation
 3: for $t = 1$ to $T$ do
 4:     Train GBM $f_t$ with tuned parameters on $\mathcal{L}(t-1)$
 5:     Apply $f_t$ to $\mathcal{U}(t-1)$ to obtain predictions $\hat{y}_u(t)$
 6:     Assign each $u \in \mathcal{U}(t-1)$ to its most likely class
 7:     Calculate each $z_u(t)$ as defined in (6.1)
 8:     Construct one IF for each class (Section 6.4.4).
 9:     Compute each $a_u(t)$ as defined in (6.3)
10:     Compose $\mathcal{Q}(t)$ using query function $\psi^{\text{Jas}}$ as described in Section 6.4.5
11:     Obtain actual classes $y_q$ of $q \in \mathcal{Q}(t)$ by human expert
12:     Use $y_q$ and predictions $\hat{y}_q(t)$ to determine $\delta_a^{\beta}(t)$, $\delta_z^{\beta}(t)$, and $\Delta^{\gamma}(t)$ with (6.4), (6.5) and (6.6), respectively.
13:     Update query fractions to obtain $\alpha_a(t+1)$ and $\alpha_z(t+1)$ (Section 6.4.6)
14:     $\mathcal{L}(t) \Leftarrow \mathcal{L}(t-1) \cup \mathcal{Q}(t)$ and $\mathcal{U}(t) \Leftarrow \mathcal{U}(t-1) \setminus \mathcal{Q}(t)$
15: end for
16: return
```

element $j \in \mathcal{J}$ is a four-dimensional vector of the form $(\alpha_a^{(0)}, \beta, \gamma, \tau)$. Let Q_J be the number of unlabelled observations that should be queried in each iteration. Ideally, the value of Q_J is close to Q, but we do want to perform some iterations before $\mathcal{L}_J(t)$ reaches the size of $\mathcal{L}(0)$. The number of iterations in Jasmine tuning is given by $T_J := \lceil \frac{U_J(0)}{Q_J} \rceil - 1$, where $U_J(0) := |\mathcal{U}_J(0)|$. The minus one is because the last unlabelled observations are the least informative, and hence, not of interest to query.

Let $j \in \mathcal{J}$ be some hyperparameter combination. Then a GBM model is trained on $\mathcal{L}_J(0)$, similar to what we described in Section 6.4.3. This model is then applied to the evaluation set $\mathcal{E}_J$ resulting in some performance metric $p_j(0)$. Then the rest of the AL procedure, as we described in Sections 6.4.4 to 6.4.7, is executed. After all iterations are performed, the sequence of performance measures $\{p_j(t-1)\}_{\{t=1,\dots,T_J\}}$ is obtained. We use this sequence to determine which hyperparameter combination works best, since such a sequence is obtained for each $j \in \mathcal{J}$.

Because stochasticity is involved, we repeat Jasmine tuning S_J times. Each simulation, the set $\mathcal{L}(0)$ is randomly divided in the sets $\mathcal{L}_J(0)$, $\mathcal{U}_J(0)$, and $\mathcal{E}_J$. Also, each simulation yields a sequence of performance metrics for each hyperparameter combination. Then the combination that yields the best performance over the simulations is taken as (relatively) optimal for $\alpha_a^{(0)}$, β, γ, and τ.

6.5 Experimental Setup

We conducted several experiments to determine whether our AL method Jasmine performed better than ALADIN and to decide if α-dynamic updating yielded significant improvements over baseline query functions. In this section, we discuss the datasets on which the experiments were performed. These sets are fully labelled, so the labels for the observations in the ‘unlabelled’ set were hidden until they were queried by Jasmine. After that, we explain the steps taken to execute the procedures. These include which hyperparameters were tuned over which ranges and how the different AL methods were evaluated.

6.5.1 Data

Yavanoglu and Aydos [127] and Ferrag et al. [28] provided overviews of publicly available security-related datasets commonly used in NID research. The sets that were discussed span from 1999 to 2018, showing that even old network datasets are still used to benchmark NID techniques. However, this is mostly due to the lack of public data, as discussed in Section 6.1. Here, firstly, we used the *NSL-KDD* dataset for assessing the considered AL methods, since it is the most used for evaluation in the field of cyber security. Secondly, we considered the *UNSW-NB15* dataset. This set is cited often too, and is much more recent than the popular NSL-KDD data. Moreover, it has several more realistic aspects, which are discussed later. Henceforth, these datasets were used to assess the performance of the discussed AL methods and to compare the obtained results for different data.

NSL-KDD

The NSL-KDD dataset was developed by Tavallaee et al. [112] and is an improvement on the KDD-Cup-99 dataset. The latter was prepared by Stolfo et al. [105] and consists of a training set and a test set. These two sets were constructed such that they do not have the same underlying distribution. Each observation in KDD-Cup-99 is made up of a 41-dimensional feature vector and an output label with the attack type. There are four global types of actual attacks and a ‘normal’ type, indicating a benign connection. Within the attack types, there can be several distinct attack scenarios. The test set contains scenarios from the training set as well as new scenarios. As mentioned before, the NSL-KDD dataset is a revision of the KDD-Cup-99 data and addresses many of the problems. What these problems are and how the data was revised is described by Tavallaee et al. [112]. Finally, the total number of training observations in NSL-KDD is 125,972 and the number of test instances is 22,544. We removed the categorical features Protocol_type, Service, Flag, and Difficulty_level, because Jasmine is based on numerical techniques. Moreover, since Jasmine expects binary output labels, all attack types obtained value 1 (the malicious class), while the normal type obtained value 0 (the benign class).

46.5% of the training instances are malicious, while 56.9% are malicious in the test set.

NSL-KDD-rand

Besides considering the NSL-KDD data as provided by Tavallaee et al. [112], we also considered the dataset we call *NSL-KDD-rand*. We constructed this set by combining the training and test set of the NSL-KDD dataset. Now, 48.1% of all observations are malicious. During the experiments, each time a new training and test set were chosen. These new sets are expected to have the same underlying distribution, in contrast to the provided training and test set of the NSL-KDD data.

UNSW-NB15

The UNSW-NB15 dataset was constructed by Moustafa and Slay [79], partially to address some problems of the NSL-KDD data. The UNSW-NB15 dataset contains 2,540,047 observations with each network connection consisting of a 47-dimensional feature vector and two output attributes. The first output is the specific attack type and the second is a binary value indicating whether the observation is benign (0) or malicious (1). Nine attack types are present in the dataset, but we only used the second output attribute, since Jasmine expects a binary output label. We reduced the number of predictive features from 47 to 36 by removing srcip, sport, dstip, dsport, proto, state, service, stcpb, dtcpb, Stime, and Ltime, because they directly determine the output label, are categorical, or have no predictive use. Furthermore, there are several missing values in UNSW-NB15 which we chose from context to be 0. Finally, 12.6% of the observations correspond to attacks. This lower fraction of malicious traffic is one of the reasons why this dataset is closer to reality than the NSL-KDD data. More information about UNSW-NB15 is given by Moustafa and Slay [79].

Although the attack balance is more realistic, there are still relatively many malicious observations. Therefore, we also constructed a dataset in which the attacks were downsampled to 1.0% to see how Jasmine performs on highly unbalanced data.

6.5.2 Experiments

Query Functions

We considered several query functions in the experiments. First of all, we regarded Jasmine with its characteristic α-dynamic query function ψ^{Jas} (as described in Section 6.4.5) as the main focus of this research (jas.main). Furthermore, we examined some simpler query functions for the Jasmine procedure: only querying anomalies (jas.anom), only querying uncertainties (jas.uncert), and only querying random observations (jas.rand). For these three query functions,

α-dynamic updating is not involved. Note that the uncertainty query approach in jas.uncert corresponds to many of the studies discussed in Section 6.2. Also, the query strategies in jas.anom and jas.rand have ties to related work. Naturally, we also considered the full ALADIN procedure of Stokes et al. [104] (ala.main), just as the incomplete Jasmine procedure with ALADIN's query function of querying anomalies and uncertainties in a fixed 50/50 split (jas.basic). Consequently, we compared Jasmine to five different AL methods.

Global Parameters

The global parameters are the variables defined before any computation took place. These include the initial size of the labelled set $L(0)$, the initial size of the unlabelled set $U(0)$, the size of the evaluation set E, and the query set size Q. Moreover, N is the maximum number of observations that were to be queried to the human expert during the process. Together with Q, the parameter N determines the total number of iterations: $T := \lfloor N/Q \rfloor$. Finally, since Jasmine is an inherently stochastic procedure, we repeated the experiments S times. This was done to analyse how our method behaved on average and in what range its performances resided. Note that, for each repetition, the sets $\mathcal{L}(0)$ and $\mathcal{U}(0)$ were newly constructed. For the NSL-KDD-rand and UNSW-NB15 datasets, the evaluation set $\mathcal{E}$ was also freshly sampled. This was not necessary for the NSL-KDD data, since $\mathcal{E}$ is a provided fixed set.

The values chosen for the global parameters for the experiments are presented in Table 6.1. As the table shows, two different values $L(0)$ were considered, because we were interested in how the initially labelled set size influenced the performance of the AL methods. Furthermore, we chose Q to be 40, as we deemed this a good balance between allowing for the query fractions to update not too erratically (needing Q to be large) and allowing for relatively small updates to the classifier (needing Q to be small). Also, we chose N to be 15,000 because we saw from exploratory studies that the models do not drastically change any more when more labels were provided. Finally, for the NSL-KDD-rand and UNSW-NB15 datasets, we chose E to be 5,000, to obtain a representative test set without using too many observations only for evaluation.

Table 6.1: Values for global parameters

Data	$L(0)$	$U(0)$	E	Q	N	S
NSL-KDD	125	125,848	22,544	40	15,000	30
NSL-KDD	250	125,723	22,544	40	15,000	30
NSL-KDD-rand	125	146,392	5,000	40	15,000	30
NSL-KDD-rand	250	146,267	5,000	40	15,000	30
UNSW-NB15	125	2,534,922	5,000	40	15,000	30
UNSW-NB15	250	2,534,797	5,000	40	15,000	30

Hyperparameter Tuning GBM

As mentioned in Section 6.4.2, good values for the GBM were found by tuning on the initially labelled set $\mathcal{L}(0)$ with k-fold cross validation. We used the study by Tama and Rhee [111] and exploratory research to determine which hyperparameters to tune and over what range to tune them. The parameters that were not selected for tuning obtained their default settings as given by the `h2o.gbm` function of the `H2O.ai` package, which we used in the `R` programming language. The values or tuning ranges of the parameters are shown in Table 6.2. Since the number of possible hyperparameter combinations is large (more than 177 thousand), a random search was performed over all combinations for a maximum of 4 hours. The combination with the best trade-off between performance metric and computation time was chosen. We considered the F_1 score as the performance metric and we chose the threshold ε to be 10^{-4}.

Table 6.2: Tuning ranges for hyperparameters in `h2o.gbm` (`csr` = 'col_sample_rate')

distribution	histogram_type	learn_rate_annealing
Bernoulli	RoundRobin	$\{0.95, 0.99, 0.999\}$
max_depth	sample_rate	ntrees
$\{6, 12, 24\}$	$\{0.60, 0.78, 1.0\}$	$\{250, 500, 1{,}000\}$
nbins	`csr`	learn_rate
$\{10, 16, 25\}$	$\{0.84, 0.92, 1.0\}$	$\{0.02, 0.05, 0.125\}$
min_rows	`csr`_per_tree	`csr`_change_per_level
$\{6, 8, 10\}$	$\{0.40, 0.64, 1.0\}$	$\{0.94, 1.0, 1.06\}$

Jasmine Tuning

The relevant hyperparameters for the Jasmine tuning phase, as we described in Section 6.4.8, are the initial anomaly query fraction $\alpha_a^{(0)}$, the FN weight factor β, the update magnitude γ, and the decrease speed in querying random instances τ. The ranges that the parameters were tuned over are shown in Table 6.3, yielding 108 possible combinations. We chose the ranges for $\alpha_a^{(0)}$, β, and γ to be symmetric around the 'unity value'. For $\alpha_a^{(0)}$, this is $\frac{1}{2}$, since then the initial number of anomalous observations was equal to the number of uncertain instances. For β, this is 1, because then the FNs and FPs were weighed equally. For γ, this is also 1, since then $\Delta^{(\gamma)}(t)$ reduced to the linear difference $\Delta(t)$.

Table 6.3: Tuning ranges for Jasmine parameters

$\alpha_a^{(0)}$	β	γ	τ	S_J
$\{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$	$\{\frac{1}{2}, 1, 2\}$	$\{\frac{1}{2}, 1, 2\}$	$\{\frac{1}{800}, \frac{1}{400}, \frac{1}{200}, \frac{1}{100}\}$	4

During Jasmine tuning, we wanted to choose Q_J as close to Q as possible, but also perform at least three iterations. Hence, we defined the tuning query size as $Q_J := \min\{U_J(0)/4, Q\}$. The initially unlabelled set size was divided by 4 ($= 3+1$), since the last iteration was not performed. This is because we deem the last unlabelled observations the least informative. Each Jasmine parameter combination j yielded a performance metric for every time step. This produced the sequence $\{p_j(t)\}_{\{t=0,\dots,T_J-1\}}$. Again, this metric was the F_1 score. After the iterations were performed, the area underneath the $(t, p_j(t))_{\{t=0,\dots,T_J-1\}}$-'curve' was calculated. This is an example of a *learning curve*, which is commonly used in the AL paradigm to assess the quality of a method [52, 100]. The larger this area, the better parameter combination j is. As there is stochasticity in the techniques used in Jasmine, the tuning phase was repeated S_J times. The combination with the largest average area was chosen as the hyperparameter setting for Jasmine in the actual AL procedure. Note that the Jasmine-specific parameters were tuned on $\mathcal{L}(0)$, and so, Jasmine did not get an unfair advantage by seeing more data in advance than the other AL methods, which did not need to execute Jasmine tuning.

Evaluation of AL Methods

To evaluate the six different AL methods, we utilised the evaluation set $\mathcal{E}$ that was set aside each simulation. The quality of the predictions of an AL method on $\mathcal{E}$ was determined by the performance metric $p(t)$ (in this case the F_1 score) for every iteration t. After some reference value t_{ref} of iterations were performed ($0 \leq t_{\text{ref}} \leq T$), a sequence of performance metrics $\{p(t)\}_{\{t=0,\dots,t_{\text{ref}}\}}$ was obtained. Similar to Jasmine tuning, we took the area $A(t_{\text{ref}})$ underneath the $(t, p(t))_{\{t=0,\dots,t_{\text{ref}}\}}$-learning curve as a measure of performance. Briefly said, the higher $A(t_{\text{ref}})$, the better the method is up to the iteration step t_{ref}. However, since there is stochasticity involved, we repeated each complete AL procedure S times. During simulation s, $\mathcal{L}(0)$ and $\mathcal{U}(0)$ were randomly chosen (for NSL-KDD-rand and UNSW-NB15 also $\mathcal{E}$ was randomly sampled) and all six procedures were provided the same initial sets. Consequently, this led to the vector $(A_s^{(1)}(t_{\text{ref}}), \dots, A_s^{(6)}(t_{\text{ref}}))$ of paired area metrics. Next, we statistically compared the area metrics of the Jasmine α-dynamic method with the metrics of the other methods. This was done by the Wilcoxon signed-rank test (with significance threshold of 0.05) to determine whether

$$H_0^{(m)}(t_{\text{ref}}) : \underset{s=1,\dots,S}{\text{median}} \left\{ A_s^{(1)}(t_{\text{ref}}) \right\} < \underset{s=1,\dots,S}{\text{median}} \left\{ A_s^{(m)}(t_{\text{ref}}) \right\} \tag{6.10}$$

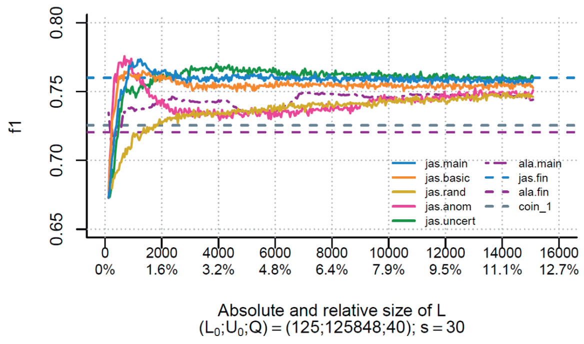
could be rejected for method $m = 2, \dots, 6$. When this was the case, then α-dynamic querying performed significantly better than the other considered query functions and AL techniques.

6.6 Results

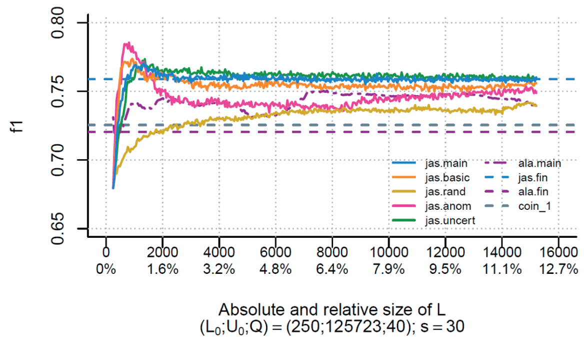
In this section, the results of our research are presented. They were obtained by performing the steps explained in Section 6.5 on the NSL-KDD, NSL-KDD-rand and UNSW-NB15 data. First of all, the learning curve of F_1 scores is shown for each of the six AL methods that we considered. This curve gives an insight in how well the classifier of a specific method performed on the evaluation set throughout the AL process. Secondly, the p-values of the Wilcoxon signed-rank test are presented for predetermined specific iteration steps. Thirdly, the dynamics of the query fractions $\alpha_a(\cdot)$, $\alpha_z(\cdot)$, and $\alpha_r(\cdot)$ are shown to illustrate how Jasmine adjusted the balance between querying anomalous, uncertain, and random observations. Finally, we discuss the implications of these results per dataset.

6.6.1 Results on NSL-KDD

Learning Curves



(a) $L(0) = 125$



(b) $L(0) = 250$

Figure 6.4: Learning curves on NSL-KDD-$\mathcal{E}$ for different initial sizes $L(0)$

Figure 6.4 shows the average learning curves for each of the six AL methods on the fixed evaluation set NSL-KDD-$\mathcal{E}$ for initially labelled set sizes $L(0) = 125$ and $L(0) = 250$. Each simulation yielded a learning curve, hence we took the average of the curves over all simulations. The blue dashed line (jas.fin) is the average final performance on $\mathcal{E}$ of the GBM trained on the complete training set with all labels available. This performance metric was $\overline{F_1} \approx 0.760$ for Figure 6.4a and $\overline{F_1} \approx 0.759$ for Figure 6.4b. The value of the line is approximately the same for both plots, because the evaluation set $\mathcal{E}$ is fixed and independent of $\mathcal{L}(0)$. However, each GBM was tuned differently, leading to small differences. Since all labels of the NSL-KDD dataset are technically available, we included the final performance to show how quickly the Jasmine-based AL methods reached this value. The purple dashed line (ala.fin) is the final performance on $\mathcal{E}$ of the logistic regression classifier of the ALADIN procedure ($F_1 \approx 0.720$). This final performance was constant during the simulations and for both settings of $L(0)$, since the classifier of ALADIN is deterministic. Finally, the grey dashed line (coin_1) is the Dutch Draw baseline on $\mathcal{E}$ ($F_1 \approx 0.725$). This baseline was obtained by classifying each evaluation observation as malicious (see Chapter 5).

Statistical Tests

Table 6.4: p-values Wilcoxon test jas.main vs. ... with $L(0) = 125$

t_{ref}	$L(t_{\text{ref}})$	jas.basic	jas.rand	jas.anom	jas.uncert	ala.main
9	485	0.992	0.000172	1.00	0.0155	0.894
16	765	0.991	$1.52 \cdot 10^{-5}$	1.00	0.0249	0.381
22	1005	0.975	$5.96 \cdot 10^{-7}$	1.00	0.00983	0.0571
34	1485	0.855	$5.00 \cdot 10^{-7}$	0.971	0.000729	0.000128
47	2005	0.786	$5.00 \cdot 10^{-7}$	0.388	0.000932	$4.99 \cdot 10^{-7}$
122	5005	0.0131	$2.99 \cdot 10^{-6}$	$1.28 \cdot 10^{-7}$	0.131	$1.30 \cdot 10^{-8}$
247	10005	$3.46 \cdot 10^{-6}$	$3.96 \cdot 10^{-5}$	$1.86 \cdot 10^{-9}$	0.908	$1.57 \cdot 10^{-7}$
372	15005	$7.10 \cdot 10^{-7}$	0.000190	$1.86 \cdot 10^{-9}$	0.975	$2.86 \cdot 10^{-7}$

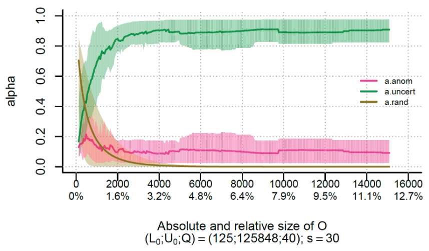
Table 6.5: p-values Wilcoxon test jas.main vs. ... with $L(0) = 250$

t_{ref}	$L(t_{\text{ref}})$	jas.basic	jas.rand	jas.anom	jas.uncert	ala.main
9	610	0.924	0.000128	0.994	0.0790	0.516
16	890	0.958	$2.57 \cdot 10^{-6}$	1.00	0.118	0.0502
22	1130	0.963	$4.99 \cdot 10^{-7}$	1.00	0.122	0.00530
34	1610	0.915	$1.28 \cdot 10^{-7}$	0.999	0.388	$3.14 \cdot 10^{-5}$
47	2130	0.897	$1.02 \cdot 10^{-7}$	0.967	0.556	$1.38 \cdot 10^{-6}$
122	5130	0.122	$4.00 \cdot 10^{-8}$	0.00233	0.997	$9.31 \cdot 10^{-9}$
247	10130	$9.43 \cdot 10^{-5}$	$4.16 \cdot 10^{-7}$	$8.20 \cdot 10^{-8}$	1.00	$8.20 \cdot 10^{-8}$
372	15130	$1.89 \cdot 10^{-6}$	$1.90 \cdot 10^{-6}$	$2.79 \cdot 10^{-9}$	1.00	$1.62 \cdot 10^{-6}$

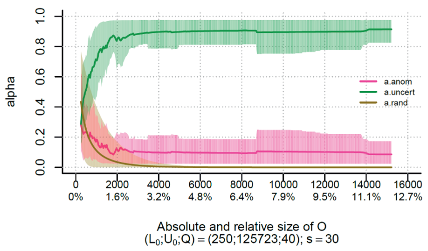
To determine whether Jasmine performed significantly better than the other five AL methods, we determined the p-value of the test in (6.10) for each method $m = 2, \ldots, 6$ and for different values of t_{ref}. Tables 6.4 and 6.5 show the results for the experiments with $L(0) = 125$ and $L(0) = 250$, respectively. A green value indicates that Jasmine (jas.main) performed significantly better than the method in the corresponding column for the labelled set size $L(t_{\text{ref}})$.

A red value, however, means that Jasmine performed significantly worse (by interchanging the sides of Equation (6.10)). A black value means that the test was indecisive and could not conclude whether Jasmine was better or worse.

α-dynamic Updating



(a) $L(0) = 125$



(b) $L(0) = 250$

Figure 6.5: Progress of query fractions for different initial sizes

Finally, Figure 6.5 presents the α-dynamic updating procedure of Jasmine for $L(0) = 125$ and $L(0) = 250$. Similar to the figure with the learning curves, every simulation resulted in an α-dynamic curve for each of the query fractions $\alpha_a(\cdot)$ (a.anom), $\alpha_z(\cdot)$ (a.uncert), and $\alpha_r(\cdot)$ (a.rand). We took the average of the query fractions to obtain the average α-dynamic curves. A shaded region indicates the interval in which 80% of the observed fractions with matching colour resided, and therefore, shows the spread of the values. Since $S = 30$ simulations were performed, each region contains the 24 'middle' fraction values.

Implication of Results on NSL-KDD

Our first observation in Figure 6.4 is that the average F_1 score rapidly increased in the first iterations for the Jasmine (jas.main), jas.anom, jas.uncert, and jas.basic procedures. This means that by using the corresponding query approaches, valuable unlabelled observations were queried to the oracle, because the GBMs were able to make better predictions on the evaluation set NSL-KDD-$\mathcal{E}$. This increase is most notable for jas.anom and then especially for $L(0) = 250$. This makes sense, because NSL-KDD-$\mathcal{E}$ contains new anomalous attack scenarios that are not found in the training set. Remarkably, the performance of jas.anom went higher than the final (average) F_1 score. For several iteration steps, also the other three methods obtained scores higher than the final performance for both settings of $L(0)$. This means that a carefully constructed smaller dataset led to better predictions on the evaluation set than the complete training set did.

Even though jas.anom performed better than the other methods at the start of the iterations, its effectiveness decreased when more anomalous observations were added to the labelled set. The decrease of effectiveness is also visible in Tables 6.4 and 6.5: for small values of t_{ref} querying only anomalies performed better than Jasmine, but later Jasmine obtained significantly better results. It could be that the labelled set became more and more abnormal when more anomalous observations were added, resulting in impaired training of the GBM classifier in later iterations after it had improved before.

Furthermore, it is clear that Jasmine performed better than only querying uncertainties, as the p-values show. It appears that also querying anomalies is important. However, when time progresses, its performance is not significantly better any more and eventually becomes significantly worse. It should be stressed, though, that the learning curves show that both jas.main and jas.uncert have converged to the final score and only differ a little.

Only querying random observations, as done by jas.rand, performed significantly worse than Jasmine for all reference iterations. This shows that specifically choosing anomalous or uncertain observations works better than only querying random observations.

Also interesting to note is that Jasmine was on par with or worse than jas.basic at the start of the iteration procedure. However, when the size of $\mathcal{L}(\cdot)$ grew, Jasmine became significantly better, as the p-values in both Tables 6.4 and 6.5 show. This means that dynamically adjusting the query balance in a later stadium has an advantage over querying anomalies and uncertainties in a fixed 50/50 fashion. Combining this with the progress of the query fractions in Figure 6.5 and with the fact that jas.anom's performance worsens over time, shows that Jasmine found the right balance between querying uncertainties and anomalies. Nonetheless, since anomalies appeared to be better at the start, it is curious why Jasmine did not query more anomalies in that stage. Hence, the information metrics as defined in (6.4) and (6.5) could have a preference for uncertain over anomalous observations.

Lastly, Jasmine performed fairly quickly significantly better than ALADIN as the p-values show. This is partly due to the simpler ML techniques in the latter, as it took Jasmine less effort to obtain better results than ALADIN than it took to perform better than jas.basic.

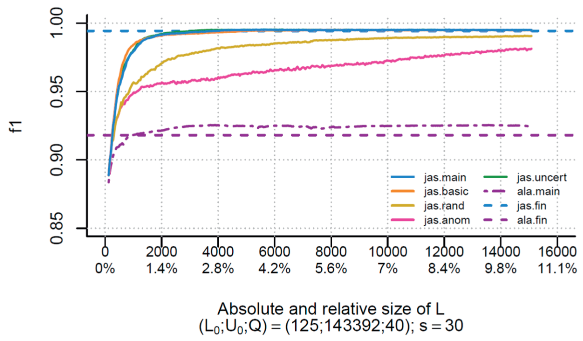
In general, there do not seem to be large differences between the experiments with $L(0) = 125$ and with $L(0) = 250$. However, the α-dynamic curves in Figure 6.5 show that the average initial random query fraction $\alpha_r(0)$ was noticeably bigger for $L(0) = 125$ than for $L(0) = 250$, since the brown curve (a.rand) starts higher in the former. This makes sense, because $\mathcal{L}(0)$ was randomly constructed, and hence, it became less essential to query random instances when we chose $L(0)$ larger.

Considering the computation times, the procedures incorporating AD (jas.main, jas.basic, and jas.anom) took significantly longer (on average 2.8 hours for $L(0) = 125$ and 3.3 hours for $L(0) = 250$) than those without the IF (jas.rand, jas.uncert, and ala.main) did (respectively 1.5, 1.8, and 2.0 hours on average for $L(0) = 125$;

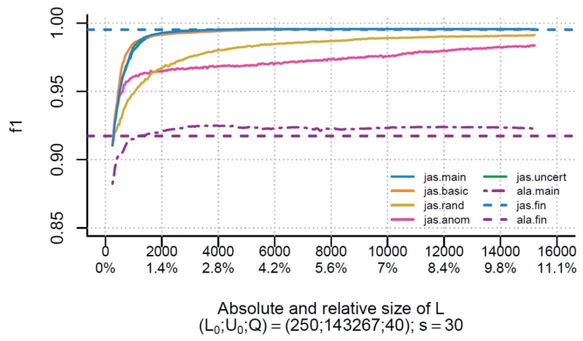
and 2.0, 2.3, and 2.1 hours for $L(0) = 250$). The longer computation times for $L(0) = 250$ compared to $L(0) = 125$ is presumably because of the hyper-parameters chosen for the GBM, e.g., more trees are constructed, thus training takes longer. This is supported by the negligible difference in times for ALADIN, which does not incorporate a GBM. It is important to note that jas.main also performs Jasmine tuning, and therefore, requires additional computation time.

6.6.2 Results on NSL-KDD-rand

Learning Curves



(a) $L(0) = 125$



(b) $L(0) = 250$

Figure 6.6: Learning curves on NSL-KDD with random evaluation set for different $L(0)$

Similar to the results presented in Section 6.6.1, Figure 6.6 presents the average learning curves for each of the six AL methods and two constant reference lines. In contrast to the results on the NSL-KDD dataset, the evaluation set $\mathcal{E}$ was chosen anew for each simulation. More specifically, at the start of each repetition the complete NSL-KDD-rand dataset was randomly partitioned in $\mathcal{L}(0)$, $\mathcal{U}(0)$, and $\mathcal{E}$. The blue dashed line (jas.fin) is the average final performance of the simulation-specific GBMs on their corresponding evaluation sets. This metric was $\overline{F_1} \approx 0.994$ for Figure 6.6a and $\overline{F_1} \approx 0.995$ for Figure 6.6b. This time, the average final performance of ALADIN represented by the purple dashed line (ala.fin) was no longer necessarily constant, but $\overline{F_1} \approx 0.918$ for initial set size $L(0) = 125$ and $\overline{F_1} \approx 0.917$ for $L(0) = 250$. Lastly, the expected performance of the best dummy classifier is not visible in the figures, since it was $\overline{F_1} \approx 0.650$ for $L(0) = 125$ and $\overline{F_1} \approx 0.651$ for $L(0) = 250$.

Statistical Tests

Table 6.6: p-values Wilcoxon test jas.main vs. ... with $L(0) = 125$

t_{ref}	$L(t_{\text{ref}})$	jas.basic	jas.rand	jas.anom	jas.uncert	ala.main
9	485	0.798	0.000365	0.0192	0.657	$5.96 \cdot 10^{-7}$
16	765	0.886	$5.96 \cdot 10^{-7}$	0.000333	0.657	$1.86 \cdot 10^{-9}$
22	1005	0.904	$1.30 \cdot 10^{-8}$	$1.52 \cdot 10^{-5}$	0.642	$1.86 \cdot 10^{-9}$
34	1485	0.894	$9.31 \cdot 10^{-10}$	$8.20 \cdot 10^{-8}$	0.672	$9.31 \cdot 10^{-10}$
47	2005	0.831	$9.31 \cdot 10^{-10}$	$9.31 \cdot 10^{-10}$	0.708	$9.31 \cdot 10^{-10}$
122	5005	0.492	$9.31 \cdot 10^{-10}$	$9.31 \cdot 10^{-10}$	0.836	$9.31 \cdot 10^{-10}$
247	10005	0.428	$9.31 \cdot 10^{-10}$	$9.31 \cdot 10^{-10}$	0.846	$9.31 \cdot 10^{-10}$
372	15005	0.365	$9.31 \cdot 10^{-10}$	$9.31 \cdot 10^{-10}$	0.841	$9.31 \cdot 10^{-10}$

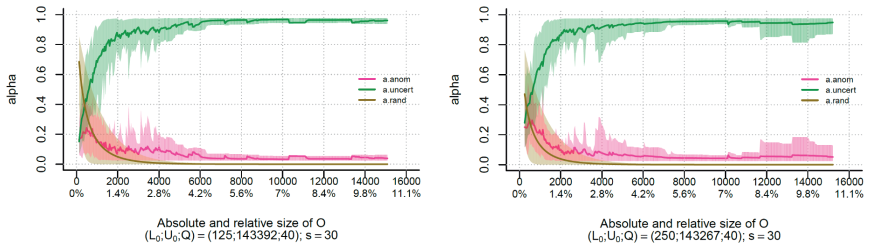
Table 6.7: p-values Wilcoxon test jas.main vs. ... with $L(0) = 250$

t_{ref}	$L(t_{\text{ref}})$	jas.basic	jas.rand	jas.anom	jas.uncert	ala.main
9	610	0.970	$4.16 \cdot 10^{-7}$	0.313	0.930	$2.79 \cdot 10^{-9}$
16	890	0.985	$1.28 \cdot 10^{-7}$	0.0261	0.701	$1.86 \cdot 10^{-9}$
22	1130	0.975	$1.77 \cdot 10^{-8}$	$8.59 \cdot 10^{-4}$	0.548	$9.31 \cdot 10^{-10}$
34	1610	0.963	$9.31 \cdot 10^{-10}$	$5.96 \cdot 10^{-7}$	0.516	$9.31 \cdot 10^{-10}$
47	2130	0.945	$9.31 \cdot 10^{-10}$	$9.31 \cdot 10^{-10}$	0.564	$9.31 \cdot 10^{-10}$
122	5130	0.635	$9.31 \cdot 10^{-10}$	$9.31 \cdot 10^{-10}$	0.650	$9.31 \cdot 10^{-10}$
247	10130	0.444	$9.31 \cdot 10^{-10}$	$9.31 \cdot 10^{-10}$	0.687	$9.31 \cdot 10^{-10}$
372	15130	0.444	$9.31 \cdot 10^{-10}$	$9.31 \cdot 10^{-10}$	0.650	$9.31 \cdot 10^{-10}$

Tables 6.6 and 6.7 show the p-values of the Wilcoxon signed-rank test with null hypothesis as given in (6.10) for different values of t_{ref}.

α-dynamic Updating

The dynamics of the α-updating procedure of Jasmine are shown in Figure 6.7 for $L(0) = 125$ and $L(0) = 250$. As described before, the curves represent the average query fractions $\alpha_a(\cdot)$ (a.anom), $\alpha_z(\cdot)$ (a.uncert), and $\alpha_r(\cdot)$ (a.rand) throughout the iteration process.



(a) $L(0) = 125$ **(b)** $L(0) = 250$

Figure 6.7: Progress of query fractions for different initial sizes

Implication of Results on NSL-KDD-rand

At first glance, the results on the randomly selected evaluation sets of NSL-KDD-rand are much better for every considered AL method. This seems reasonable, because the fixed evaluation set for the NSL-KDD data contains unseen cyber attacks on which the classifier could not train. In the case of NSL-KDD-rand, the evaluation set was randomly chosen from the complete dataset, so we expected it to have the same structure as the training set, making it easier for the classifier to learn. This was specifically the case for the GBM, because it rapidly obtained an almost perfect F_1 score on the evaluation set.

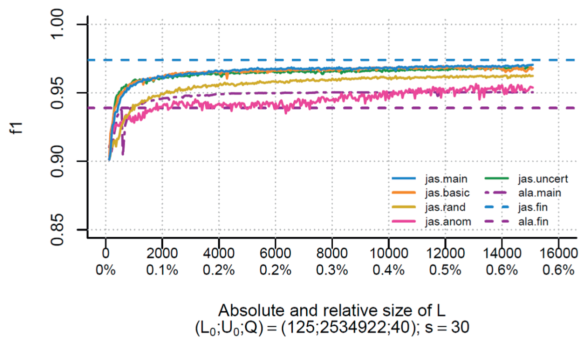
The figures also show that the three learning curves for Jasmine, jas.basic and jas.uncert increased similarly at the start of the procedure. This was not the case for jas.anom, indicating that it was mostly important to query uncertain observations. This was also reflected in the p-values, as shown by Tables 6.6 and 6.7. Jasmine quickly performed significantly better than jas.anom, but it had more difficulty in obtaining significantly better results than jas.basic and jas.uncert. The latter was even significantly better than Jasmine. However, it should be noted that the F_1 scores were already near perfect for these three query approaches. Moreover, the dynamics of the query fractions in Figure 6.7 show that there was a clear preference for querying more uncertainties than anomalous observations.

Furthermore, the learning curves show that Jasmine obtained better results than jas.rand and ALADIN. Both Tables 6.6 and 6.7 indicate that Jasmine was significantly better for all considered reference values.

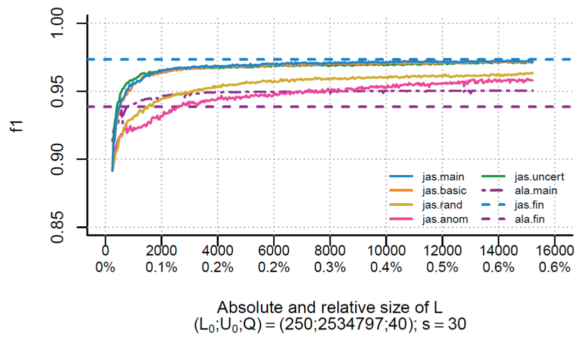
6.6.3 Results on UNSW-NB15

Learning Curves

Figure 6.8 shows the average learning curves for every one of the six AL methods and two constant reference lines. The evaluation set $\mathcal{E}$ was chosen anew for each simulation, equivalent to what we discussed in Section 6.6.2. The blue dashed line (jas.fin) is the average final performance of the simulation-specific GBMs



(a) $L(0) = 125$



(b) $L(0) = 250$

Figure 6.8: Learning curves on UNSW-NB15 with random evaluation set for different $L(0)$

on their corresponding evaluation sets. For Figure 6.8a, this metric was $\overline{F_1} \approx 0.974$, while it was $\overline{F_1} \approx 0.973$ for Figure 6.8b. The average final performance of ALADIN indicated by the purple dashed line (ala.fin) was $\overline{F_1} \approx 0.939$ for both initial set sizes. Lastly, the figures do no show the expected performance of the best dummy classifier, since it was $\overline{F_1} \approx 0.225$ for $L(0) = 125$ and $\overline{F_1} \approx 0.222$ for $L(0) = 250$. These remarkably lower baseline values are because relatively far fewer positive observations are present in UNSW-NB15 compared to NSL-KDD.

Statistical Tests

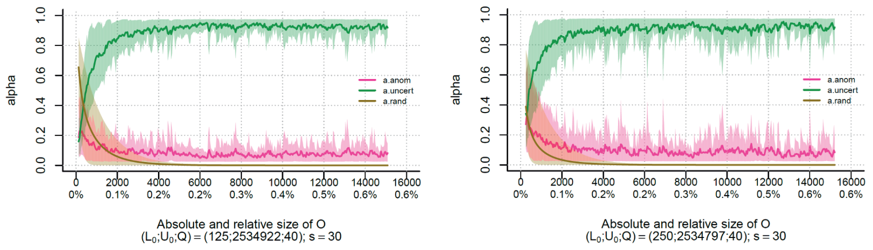
Tables 6.8 and 6.9 present the p-values of the Wilcoxon signed-rank test with null hypothesis as given in (6.10) for different values of t_{ref}.

Table 6.8: p-values Wilcoxon test jas.main vs. ... with $L(0) = 125$

t_{ref}	$L(t_{\text{ref}})$	jas.basic	jas.rand	jas.anom	jas.uncert	ala.main
9	485	0.994	$4.95 \cdot 10^{-5}$	0.0213	0.994	0.0701
16	765	0.990	$1.90 \cdot 10^{-6}$	$3.96 \cdot 10^{-5}$	0.997	$6.87 \cdot 10^{-5}$
22	1005	0.985	$9.96 \cdot 10^{-7}$	$4.00 \cdot 10^{-6}$	0.992	$3.46 \cdot 10^{-6}$
34	1485	0.965	$1.93 \cdot 10^{-7}$	$1.28 \cdot 10^{-7}$	0.971	$1.30 \cdot 10^{-8}$
47	2005	0.918	$4.00 \cdot 10^{-8}$	$6.52 \cdot 10^{-9}$	0.897	$9.31 \cdot 10^{-9}$
122	5005	0.815	$9.31 \cdot 10^{-9}$	$1.86 \cdot 10^{-9}$	0.612	$2.79 \cdot 10^{-9}$
247	10005	0.271	$5.12 \cdot 10^{-8}$	$1.86 \cdot 10^{-9}$	0.191	$1.86 \cdot 10^{-9}$
372	15005	0.122	$1.28 \cdot 10^{-7}$	$1.86 \cdot 10^{-9}$	0.131	$1.86 \cdot 10^{-9}$

Table 6.9: p-values Wilcoxon test jas.main vs. ... with $L(0) = 250$

t_{ref}	$L(t_{\text{ref}})$	jas.basic	jas.rand	jas.anom	jas.uncert	ala.main
9	610	0.715	$3.46 \cdot 10^{-7}$	$7.99 \cdot 10^{-6}$	0.989	0.149
16	890	0.735	$5.96 \cdot 10^{-7}$	$1.90 \cdot 10^{-6}$	0.998	0.00128
22	1130	0.761	$4.16 \cdot 10^{-7}$	$2.36 \cdot 10^{-7}$	0.997	0.000230
34	1610	0.650	$9.31 \cdot 10^{-9}$	$1.77 \cdot 10^{-8}$	0.998	$1.52 \cdot 10^{-5}$
47	2130	0.350	$2.79 \cdot 10^{-9}$	$9.31 \cdot 10^{-10}$	0.990	$2.57 \cdot 10^{-6}$
122	5130	0.185	$9.31 \cdot 10^{-10}$	$9.31 \cdot 10^{-10}$	0.786	$9.31 \cdot 10^{-10}$
247	10130	0.0481	$9.31 \cdot 10^{-10}$	$9.31 \cdot 10^{-10}$	0.306	$9.31 \cdot 10^{-10}$
372	15130	0.0275	$9.31 \cdot 10^{-10}$	$9.31 \cdot 10^{-10}$	0.180	$9.31 \cdot 10^{-10}$



(a) $L(0) = 125$ **(b)** $L(0) = 250$

Figure 6.9: Progress of query fractions for different initial size $L(0)$

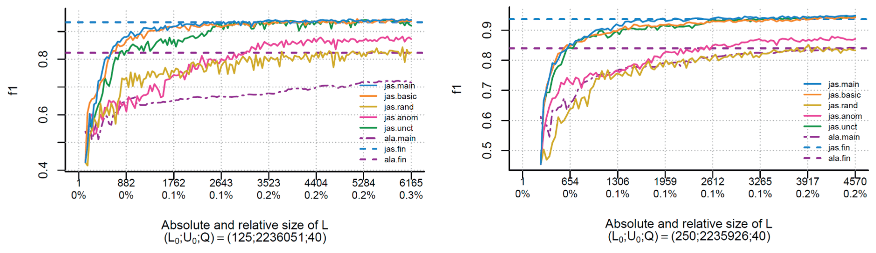
α-dynamic Updating

Figure 6.9 shows the α-dynamic updating procedure of Jasmine for $L(0) = 125$ and $L(0) = 250$. Equivalent to the results in Sections 6.6.1 and 6.6.2, the curves indicate the average query fractions $\alpha_a(\cdot)$ (a.anom), $\alpha_z(\cdot)$ (a.uncert), and $\alpha_r(\cdot)$ (a.rand) throughout the iteration process.

Implication of Results on UNSW-NB15

There are no striking differences between the two plots in Figure 6.8. However, it seems that the average learning curves for initial set size $L(0) = 125$ are a bit more shaky. Since $L(0)$ is the only different global parameter between the two plots, the change in behaviour is presumably due to how the GBM parameters and Jasmine-specific parameters were tuned. It probably also had to do with the imbalance of this dataset. Approximately 12.6% of the data is related to cyber attacks, so on average about 16 observations were malicious in $\mathcal{L}(0)$. For tuning purposes, this initial set was split in training, validation, and test sets, making it possible that only one malicious observation ended up in any of those sets. Consequently, the tuned parameters possibly led to less stable behaviour of the GBM and the α-dynamic update procedure. Hence, balancing techniques of the training data such as under- or oversampling could be considered to improve stability.

Furthermore, the learning curves in Figure 6.8 are similar to those for NSL-KDD-rand. This is also true for the p-values presented in Tables 6.8 and 6.9, and the dynamics of the average α-update curves shown in Figure 6.9.



(a) $L(0) = 125$ **(b)** $L(0) = 250$

Figure 6.10: Learning curves on unbalanced version of UNSW-NB15 for different $L(0)$

Additionally, the precise effect of the attack imbalance was studied by randomly downsampling the malicious observations in UNSW-NB15 from 12.6% to 1.0%. Figure 6.10 shows the results of the experiments on this highly unbalanced dataset. The learning curves (averaged over 20 runs) show that all procedures struggled at the start of the labelling process: performance was much lower than in Figure 6.8. However, jas.main, jas.basic, and jas.uncert improve very quickly. Moreover, Jasmine is usually the best performing procedure, and for the uncommon cases it is not, it appears to be second best.

6.7 Discussion

In this final section of the chapter, we firstly draw conclusions based on the implications of the results discussed in Section 6.6. Secondly, we consider further directions for AL in the field of NID.

6.7.1 Conclusion

The goal of this research was to propose our hybrid AL method Jasmine for NID. It consists of α-dynamic querying, which means Jasmine is able to dynamically adjust the balance between querying anomalous, uncertain, and random observations. Consequently, only the potentially most interesting observations are presented to the human expert. This sets our method apart from other AL approaches that have a static query function.

On the datasets that we considered, Jasmine performed significantly better than ALADIN, the benchmark AL method used in this research. This means for practitioners that it is beneficial to choose Jasmine over ALADIN for their AL problems. However, we should note that Jasmine needs preprocessing time, which ALADIN and other static AL methods do not need. This is because of GBM tuning and Jasmine tuning. Furthermore, we observed that Jasmine performed more robustly with respect to the datasets. We compared the characteristic α-dynamic query function of Jasmine with other query functions: querying only anomalies, only uncertainties, only random observations, and querying anomalous and uncertain observations in a fixed 50/50 fashion. We noticed that Jasmine did not always outperform the other approaches, but its performance was more stable over the different data settings, and hence, more robust. On the NSL-KDD dataset, only querying anomalies performed better when the labelling process had just started, but Jasmine took over in the long run. On the NSL-KDD-rand and UNSW-NB15 data, only querying uncertainties or querying in a 50/50 fashion reigned supreme, but Jasmine followed closely. Only querying anomalies was clearly a bad strategy for these two datasets. These findings suggest that Jasmine is better able to adapt to the provided labelled dataset $\mathcal{L}(\cdot)$. This is particularly interesting in the face of concept drift: NIDSs operate in high non-stationary contexts, which makes it wise to consider an AL method that is able to adjust its query approach dynamically. Lastly, we saw that the performance of Jasmine is not hindered by unbalanced data. Initially it does perform worse, but it quickly improves when the labelling procedure starts and it becomes the best or one of the best performing methods.

However, there is room for improvement in the way α-dynamic querying is performed here. The results on NSL-KDD show that at first querying more anomalous than uncertain observations was beneficial, but this was not reflected in the way that the query fractions were updated. The updating procedure seems to have a bias towards querying uncertainties.

6.7.2 Future Work

Our first suggestion for further research is to reconsider the α-dynamic query process such that the bias towards selecting uncertain observations is eliminated.

Another suggestion is to add unlabelled observations about which the classifier has a high prediction certainty to the labelled set without asking the oracle for its label. Consequently, the labelled set increases much more during an iteration, while the human expert does not have to label more observations. This reduces labelling time drastically and possibly leads to better predictions in an earlier stage of the AL process. We elaborate on this concept in Chapter 7. Moreover, when more labels become available, the preprocessing steps can be repeated (in a less extensive version) to allow for the GBM to better adjust to the changing training set by retuning its hyperparameters. Also, some Jasmine-specific parameters could be retuned during the process.

Our last suggestion is to consider human uncertainty in the AL method, since it is not always clear whether a connection is benign or malicious. This can be done by asking the oracle for their confidence in each label that they provide or by incorporating a general probability.

These suggestions would increase the deployment efficiency of Jasmine even more. Therefore, we would like to apply our method in a practical setting to see how it performs.

6.8 Appendix

In this section, we provide the mathematical specifications about α-dynamic querying. Before we can give the explicit definitions of the query fractions for the next iteration, we have to derive some general restrictions on the values of the fractions $\alpha_r(t)$, $\alpha_a(t)$, and $\alpha_z(t)$ for all $t \in \{0, \dots, T\}$, where T denotes the maximum number of iterations. First, a fundamental requirement of the fractions is that

$$\alpha_r(t) + \alpha_a(t) + \alpha_z(t) = 1, \tag{6.11}$$

and $\alpha_r(t), \alpha_a(t), \alpha_z(t) \in [0, 1]$. Furthermore, each iteration, we want to query at least one anomaly and one uncertainty, otherwise (6.4) or (6.5) is undefined, and consequently, $\Delta^\gamma(t)$ is undefined. Thus, we need $\alpha_a(t)$ and $\alpha_z(t)$ to be at least $\alpha_{a,z}^{\min} := 1/Q$. Then, the number of anomalies and uncertainties in $\mathcal{Q}(t)$ is at least $Q \cdot \alpha_{a,z}^{\min} = 1$. This means the upper bounds for $\alpha_a(t)$ and $\alpha_z(t)$ are restricted to be at most $1 - \alpha_{a,z}^{\min}$. The upper bound for $\alpha_r(t)$ is at most $\alpha_r^{\max} := 1 - 2\alpha_{a,z}^{\min}$, since we do allow to query no random observations, and hence, allow $\alpha_r(t)$ to go to 0. The definition of $\alpha_r(t)$ is given in (6.9). Note that $\alpha_r(t) \in [\alpha_r^{\min}, \alpha_r^{\max}) \subset [0, 1]$ with $\alpha_r^{\min} := \alpha_r(T)$. Now, we can define the upper bound for $\alpha_a(t)$ and $\alpha_z(t)$ as

$$\alpha_{a,z}^{\max}(t) := 1 - \alpha_r(t) - \alpha_{a,z}^{\min}.$$

Note that this upper bound depends on the iteration number t.

By using *(i)* $\alpha_r(t), \alpha_a(t), \alpha_z(t) \in [0, 1]$, *(ii)* (6.11), and *(iii)* the definitions of $\alpha_{a,z}^{\min}$, $\alpha_r^{\min}$, $\alpha_{a,z}^{\max}(t)$, and $\alpha_r^{\max}$, we characterise the w variables introduced in update rules (6.7) and (6.8) as follows:

$$\begin{aligned}
w_a^{(1)}(t) &= \alpha_{a,z}^{\max}(t) - \alpha_a(t)\\
w_a^{(2)}(t) &= \alpha_a(t) - \alpha_{a,z}^{\min}\\
w_z^{(1)}(t) &= \alpha_{a,z}^{\max}(t) - \alpha_z(t)\\
w_z^{(2)}(t) &= \alpha_z(t) - \alpha_{a,z}^{\min}.
\end{aligned}$$

Let us explain the specifics of the update of the anomaly fraction $\alpha_a(\cdot)$. We take the old value of $\alpha_a(t)$ as a starting point for $\alpha_a(t+1)$. This fraction can be increased by at most $w_a^{(1)}(t) = \alpha_{a,z}^{\max}(t) - \alpha_a(t)$ to obtain $\alpha_{a,z}^{\max}(t)$ as the new value. This increase $w_a^{(1)}(t)$ is scaled down by $\max\{0, \Delta^\gamma(t)\}$ such that the increment is proportional to the value of $\Delta^\gamma(t)$. Similarly, $\alpha_a(t)$ can be decreased by at most $w_a^{(2)}(t) = \alpha_a(t) - \alpha_{a,z}^{\min}$ to obtain $\alpha_{a,z}^{\min}$. The decrease $w_a^{(2)}(t)$ is then scaled down by $\min\{0, \Delta^\gamma(t)\}$. Hence, the constants ensure that $\alpha_a(t+1)$ lies in the interval $I_{a,z}(t) := [\alpha_{a,z}^{\min}, \alpha_{a,z}^{\max}(t)]$. However, it should lie in the interval $I_{a,z}(t+1) := [\alpha_{a,z}^{\min}, \alpha_{a,z}^{\max}(t+1)]$. This is why we apply the linear transformation $\lambda_{t+1} : I_{a,z}(t) \to I_{a,z}(t+1)$. This function is given by

$$\lambda_{t+1}(\alpha) = \frac{\alpha_{a,z}^{\max}(t+1) - \alpha_{a,z}^{\min}}{\alpha_{a,z}^{\max}(t) - \alpha_{a,z}^{\min}}(\alpha - \alpha_{a,z}^{\min}) + \alpha_{a,z}^{\min}. \tag{6.12}$$

Note that this function is not well-defined whenever $\alpha_{a,z}^{\max}(t) - \alpha_{a,z}^{\min} = 0$. This happens when $\alpha_r(t) = 1 - \alpha_{a,z}^{\min} = \alpha_r^{\max}$. However, the definition of $\alpha_r(t)$ in Equation (6.9) shows that $\alpha_r(t)$ is strictly smaller than $\alpha_r^{\max}$, and hence the denominator in Equation (6.12) cannot be zero and λ_{t+1} is well-defined.

Finally, for $t \in \{0, \ldots, T-1\}$, the systems of equations for the three query fractions are given by

$$\begin{cases}
\alpha_r(0) = \alpha_r^{\max} \cdot 2^{-\tau \cdot L(0)}\\
\alpha_r(t+1) = \alpha_r^{\max} \cdot 2^{-\tau \cdot (L(0) + Q \cdot (t+1))},
\end{cases}$$

$$\begin{cases}
\alpha_a(0) = \alpha_a^{(0)}\\
\alpha_a(t+1) = \lambda_{t+1}\Big(\alpha_a(t) + (\alpha_{a,z}^{\max}(t) - \alpha_a(t)) \max\{0, \Delta^\gamma(t)\}\\
\qquad\qquad\qquad + (\alpha_a(t) - \alpha_{a,z}^{\min}) \min\{0, \Delta^\gamma(t)\}\Big),
\end{cases}$$

and

$$\begin{cases}
\alpha_z(0) = \alpha_z^{(0)}\\
\alpha_z(t+1) = \lambda_{t+1}\Big(\alpha_z(t) + (\alpha_{a,z}^{\max}(t) - \alpha_z(t)) \max\{0, -\Delta^\gamma(t)\}\\
\qquad\qquad\qquad + (\alpha_z(t) - \alpha_{a,z}^{\min}) \min\{0, -\Delta^\gamma(t)\}\Big).
\end{cases}$$

7 Plusmine: Dynamic Active Learning with Semi-Supervised Learning for Automatic Classification

Hoe weet je dat dan allemaal?

Sanne Vogel
Wie is de Mol? 2010

A major problem in cyber security research is the correct labelling of up-to-date datasets. It relies on the availability of human experts, and is as such very cumbersome. Motivated by this, two techniques have been proposed for efficient labelling: Active Learning (AL) and Semi-Supervised Learning (SeSL). In this chapter, we introduce Plusmine: a network intrusion detection method that combines the benefits of AL and SeSL to efficiently automate classification. We develop new techniques for both components. Firstly, the query approach in the AL component is based on Jasmine, but we make an important improvement to its dynamic updating. Secondly, the SeSL component considers the consequences of virtually labelling candidate observations to determine which of them will actually be automatically classified. Moreover, we empirically show that Plusmine obtains good and more robust results than benchmark methods.

Based on [46]:
Jan Klein, Sandjai Bhulai, Mark Hoogendoorn, and Rob van der Mei
Plusmine: Dynamic Active Learning with Semi-Supervised Learning for Automatic Classification
2021 IEEE/WIC/ACM International Conference on Web Intelligence

7.1 Introduction

Cyber crime has become one of the most influential forms of criminality. Cybersecurity Ventures predicted the global damage to be $6 trillion USD in 2021 and estimated this to increase to $10.5 trillion in the year 2025 [75]. The already rapid growth of cyber crime has been accelerated by the outbreak of the COVID-19 pandemic in early 2020, since the dependency on online communication became even larger. Fortunately, in academia, the interest in cyber security research has also grown [77]. Most studies are focused on protecting networks of computers by means of a Network Intrusion Detection System (NIDS) [101]. Although there are several well-known problems of NIDSs, such as either a high false positive rate or high false negative rate [107], Xin et al. state that there are more significant issues [124]. Most notably, there is a lack of realistic, up-to-date cyber data, which makes it challenging to replicate results of newly developed techniques in actual computer network settings. This lack is caused by two problems: *(i)* issues in security and privacy, and *(ii)* difficulties in labelling network observations for training purposes, because connections are diverse, dynamic, and with many [103].

To mitigate the second problem, the labelling procedure of a cyber expert could be optimised such that they only need to classify the connections that are expected to increase overall detection performance the most. After labelling and adding these observations, the model can retrain itself on the increased dataset and select the next batch of interesting connections. This description characterises *Active Learning* (AL). Although AL is an efficient approach to accelerate learning, it is still restricted by the labelling speed of the cyber analyst, since the time it takes to correctly classify a network connection can fluctuate drastically [39]. It is, therefore, beneficial if the method itself could exploit the rich set of unlabelled data and could automatically classify specific observations. This is where *Semi-Supervised Learning* (SeSL) comes into play. Each iteration, a subset of observations is specifically selected for Automatic Classification (AC) and is then added to the labelled set. For example, a connection is automatically labelled when the confidence of the model in its prediction exceeds some threshold [115]. SeSL techniques allow the network intrusion detection (NID) method to quickly expand the labelled pool and increase performance without directly questioning the cyber analyst.

In this chapter, we propose a novel NIDS called *Plusmine* that combines AL and SeSL. Our *Active Semi-Supervised Learning* (ASeSL) method consists of an improved state-of-the-art AL component and a new transductive SeSL approach that considers the expected consequences of labelling observations in the next time step and uses this to determine which connections are the best candidates for AC. This is in contrast to other ASeSL methods for NID [69, 71, 89, 115, 133]. Our AC strategy is both powerful and simple. The AL component of Plusmine is an improved version of the Jasmine method that was introduced in Chapter 6. Jasmine incorporates a dynamic query function that allows the model to learn the best query approach during the labelling process. This makes it an adaptable and robust method. However, we make an important improvement to Jasmine to fix the bias it has towards querying a certain type of observations. We apply Plusmine to two popular NID datasets that we use in four dataset configurations. We demonstrate that Plusmine obtains good, more robust, and more reliable results compared to benchmark methods.

The following parts of this chapter are structured as follows. Literature on AL and ASeSL is reviewed in Section 7.2. In Section 7.3, we provide a summary of the workings of Jasmine and introduce mathematical notation. Section 7.4 proposes the Plusmine methodology: improvements to Jasmine are discussed and AC is introduced. The experiments that were performed to show the benefits of Plusmine are explained in Section 7.5. The results are given and discussed in Section 7.6. Finally, we draw conclusions in Section 7.7 and provide directions for future research.

7.2 Related Work

7.2.1 Active Learning

Active Learning is a type of Machine Learning (ML) in which the learner can interactively query an oracle to classify certain unlabelled observations. An extensive explanation on all intricacies of the AL paradigm is provided in Section 6.2.

Network Intrusion Detection

Section 6.2.2 gives an extensive literature review on the AL techniques that have been proposed in cyber security research. However, as with all AL methods, the only source of labels is the oracle, and hence, the model is limited by the labelling effort. Therefore, our method Plusmine incorporates SeSL to perform Automatic Classification without human intervention.

7.2.2 Active Semi-Supervised Learning

SeSL is in between supervised and unsupervised learning. It extends one of the two types by using information from the other [138]. For example, Levatić et al. [57] use tree-based learning that exploits both labelled and unlabelled data to improve performance. In our research, we combine SeSL with AL.

Combining Paradigms

There are several ways to combine the AL and SeSL paradigms. On the one hand, they can operate relatively independently. Tomanek and Hahn propose an approach in which highly uncertain observations are queried to the oracle while highly confident ones are labelled automatically [115]. The authors show that their approach reduces the labelling efforts by 60% compared to only AL when the right confidence threshold is chosen. Leng et al. introduce an AL method with a Support Vector Machine that queries uncertain instances to the oracle, while it uses different SeSL techniques to automatically classify observations about which the model is confident [56]. They show that some ASeSL methods perform better than using only AL, while others perform worse. This demonstrates that the inclusion of SeSL can also have a negative effect. Hady and Schwenker consider the same perspective for AL and SeSL. They remark that both learning paradigms "tackle the same problem, but from different directions" [40].

On the other hand, SeSL and AL can be intertwined more, e.g., SeSL can be made part of the query strategy: its results directly determine which observations are presented to the oracle. Zhu et al. estimate the expected classification error after an observation is queried. This leads to a better query strategy than selecting the most uncertain instances [139]. Here, both learning paradigms are again used to tackle the same problem, but now together from the same direction.

Network Intrusion Detection

Although promising results have been obtained for ASeSL techniques, they have not been explored much in cyber security research. Both Mao et al. and Zhang et al. introduce methods that combine AL with co-training [69, 133]. Co-training is used for binary classification when the features can be separated in two uncorrelated sets or *views* that are both sufficient for learning, meaning each view separately is enough for classification. Although both methods obtain good results, Zhang et al. mention it is necessary that the sufficiency and uncorrelatedness assumptions hold, which is not easy to achieve in practice. Meng and Kwok confirm this: they state lots of human efforts are necessary to obtain two uncorrelated, sufficient feature sets [71].

Because of the aforementioned limitations, our method Plusmine does not consider a multi-view approach. In fact, it uses a novel SeSL strategy in which the observations are automatically classified when they are expected to contribute the most to the intrusion detector in the next time step. Although it has similarities with the non-cyber research of Zhu et al. [139], they use the consequences of potentially labelling observations for their query strategy and not for AC, as we do. Besides this, the AL and SeSL paradigms are not intertwined in Plusmine. This makes it easier to analyse the contribution of each component separately.

7.3 Preliminaries

Before we can delve into the details of our method Plusmine, it is necessary to know the workings of Jasmine and the relevant mathematical notations. Jasmine is used as the basis for the AL component of Plumsine. Therefore, we refer the reader to Sections 6.3 and 6.4 for a detailed explanation. Here, we provide a summarised version.

Following the general AL framework, Jasmine trains a classifier on the labelled set $\mathcal{L}(t-1)$ during iteration $t \in \{1, \ldots, T\}$. If the classifier has hyperparameters, they are tuned beforehand on $\mathcal{L}(0)$. Also, Jasmine-specific parameters are tuned on this. The trained classifier is applied to the unlabelled set $\mathcal{U}(t-1)$ to obtain the malicious probability of each observation u and its predicted class. This also yields uncertainty scores $z_u(t)$, one of the two *informativeness measures* in Jasmine, for all $u \in \mathcal{U}(t-1)$. An informativeness measure is used to determine which observations are queried. Simultaneously, an anomaly detector is constructed for each class. These detectors yield the anomaly scores $a_u(t)$, the second informativeness measure, for all unlabelled observations. Now, the Jasmine-specific query function constructs the query set $\mathcal{Q}(t) \subset \mathcal{U}(t-1)$ by using the uncertainty and anomaly scores. This set is of fixed size $Q \in \mathbb{N}$ and a mix of uncertain, anomalous, and randomly selected observations according to the query type fractions $\alpha_z(t)$, $\alpha_a(t)$, and $\alpha_r(t)$, respectively. Then, the oracle provides the actual response values $y_q(t) \in \{0, 1\}$ of the query items $q \in \mathcal{Q}(t)$. Note that the labels of these observations were unknown (thus had value '$*$') in the previous time steps. The queried instances are added to the labelled pool to obtain $\mathcal{L}(t)$ and removed from the unlabelled set to obtain $\mathcal{U}(t)$. Next, the query type fractions are updated. To this end, the *anomaly information metric* $\delta_a^\beta(t)$ (see Equation (6.4)) and *uncertainty information metric* $\delta_z^\beta(t)$ (see Equation (6.5)) are calculated. In short, they measure how much information each query type adds on average. When $\delta_a^\beta(t) > \delta_z^\beta(t)$, more anomalies are queried next iteration, while more uncertainties are queried when $\delta_a^\beta(t) < \delta_z^\beta(t)$. This update procedure is called *α-dynamic updating* and is the key component of Jasmine. Finally, the time step is increased and the procedure is repeated.

7.4 Methodology

In this section, we propose Plusmine. Its AL component is based on Jasmine. However, we make some crucial changes to its query approach to eliminate the bias towards querying uncertain observations. After that, the SeSL component of Plusmine that constitutes Automatic Classification is introduced.

7.4.1 Improvements over Jasmine

Construction of Query Set

First of all, the construction of the query set $\mathcal{Q}(t)$ (as explained in Section 6.4.5) is adjusted. In Jasmine, it is enforced that the number of predicted benign and malicious observations is equal within both the anomalous and uncertain query types (if possible). In Plusmine, this restriction is relaxed such that only at least one observation of each class is necessary (if possible). This is because we do not want to force a 50/50 split in $\mathcal{Q}(t)$ for imbalanced datasets.

α-dynamic Updating

Secondly, the unique update procedure of Jasmine called α-dynamic updating (see Section 6.4.6) is improved. In Plusmine, the information metrics are changed to address the bias towards querying uncertain observations. They become

$$\begin{aligned} \delta_a^\beta(t) &:= \frac{\sum_{q\in\mathcal{Q}_a(t)} (\tilde{z}_q(t)+\tilde{a}_q(t))\cdot\left(1+(\beta-1)\mathbf{1}_{\{y_q(t)=1\}}\right)}{2\,(Q_a(t)+(\beta-1)\cdot|\{q\in\mathcal{Q}_a(t):y_q(t)=1\}|)}, \\ \delta_z^\beta(t) &:= \frac{\sum_{q\in\mathcal{Q}_z(t)} (\tilde{z}_q(t)+\tilde{a}_q(t))\cdot\left(1+(\beta-1)\mathbf{1}_{\{y_q(t)=1\}}\right)}{2\,(Q_z(t)+(\beta-1)\cdot|\{q\in\mathcal{Q}_z(t):y_q(t)=1\}|)}, \end{aligned} \tag{7.1}$$

with $\mathcal{Q}_a(t)$ and $\mathcal{Q}_z(t)$ the subsets of anomalous and uncertain observations in $\mathcal{Q}(t)$ with sizes $Q_a(t)$ and $Q_z(t)$, respectively. Note that $\mathbf{1}_{\{.\}}$ represents the indicator function. To explain the information metrics, consider Equation (7.1). The term $\tilde{z}_q(t)+\tilde{a}_q(t)$ is the sum of the normalised uncertainty score $\tilde{z}_q(t)$ and normalised anomaly score $\tilde{a}_q(t)$, and indicates how informative observation $q\in\mathcal{Q}_a(t)$ is. The larger the scores, the more information the sum conveys. In Jasmine, the anomaly score was not taken into account, only the uncertainty score (see Equations (6.4) and (6.5)). Therefore, the bias towards querying uncertainties is expected to be fixed in Plusmine. Also, the parameter β puts more ($\beta>1$) or less ($0<\beta<1$) emphasis on a malicious observation compared to a benign one, i.e., a false negative weighs respectively more or less than a false positive. Finally, the denominator ensures that $\delta_a^\beta(t), \delta_z^\beta(t)\in[0,1]$.

7.4.2 Semi-Supervised Learning

Besides labelling by a human oracle, Plusmine uses transductive SeSL to assign labels to observations. To this end, our method constructs the set of candidate observations $\mathcal{A}(t) := \mathcal{U}(t-1)\setminus\mathcal{Q}(t)$ for Automatic Classification. The instances in the query set $\mathcal{Q}(t)$ are clearly not appropriate, because they obtain the real label by the oracle. Now, $S\in\mathbb{N}$ disjoint subsets of size $B\in\mathbb{N}$ are drawn without replacement from $\mathcal{A}(t)$ to obtain $\mathcal{B}_1(t),\ldots,\mathcal{B}_S(t)$. The observations in these subsets have their predicted classes (as is often done [126]). Then, for all $s\in\{1,\ldots,S\}$, a decision tree is trained on a random subset of $\mathcal{L}(t-1)\cup\mathcal{Q}(t)\cup\mathcal{B}_s(t)$. Note that $\mathcal{L}(t-1)$ and $\mathcal{Q}(t)$ have their real labels, while $\mathcal{B}_s(t)$ has predicted labels. Next, the classifier is validated on the remaining set of observations, which yields the performance metric $V_s(t)$. The set $\mathcal{B}_{\bar{s}}(t)$ with $\bar{s} := \arg\max_{s\in\{1,\ldots,S\}}\{V_s(t)\}$ is selected as the (relatively) optimal subset to be automatically classified (assuming that a larger $V_s(t)$ means better performance). Lastly, both $\mathcal{Q}(t)$ and $\mathcal{B}_{\bar{s}}(t)$ are added to $\mathcal{L}(t-1)$ to obtain $\mathcal{L}(t)$ and removed from $\mathcal{U}(t-1)$ to obtain $\mathcal{U}(t)$. The complete Plusmine method is illustrated in Figure 7.1.

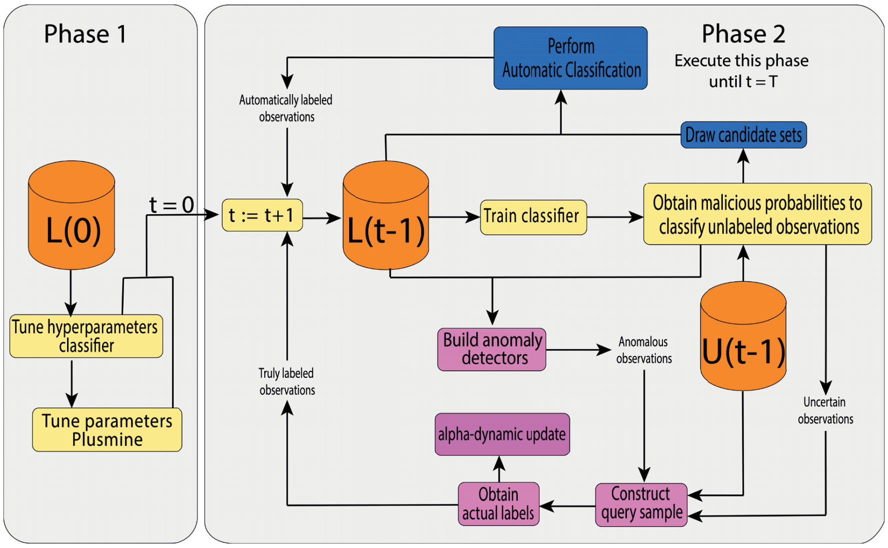


Figure 7.1: Illustration of Plusmine methodology

AC-specific Parameters

In general, the larger the number of subsets S is, the higher the likelihood is that a good candidate set for AC is found. However, a larger S means a higher computation time too. The size of each subset B is also a parameter that should be selected beforehand. We consider this a Plusmine-specific hyperparameter and is therefore tuned on $\mathcal{L}(0)$ in the same way the hyperparameters for the AL component of Plusmine are tuned. This tuning is explained in Section 7.5.2.

7.5 Experimental Setup

We conducted several experiments to demonstrate whether Plusmine performs better than benchmark methods.

7.5.1 Data

Ferrag et al. present an overview of datasets that are commonly used in NID research ranging from 1999 to 2018 [28]. From this overview, we selected the popular *NSL-KDD* data and more recent *UNSW-NB15* dataset for our investigations.

NSL-KDD

The NSL-KDD dataset was published in 2009 and is partitioned into a fixed training and evaluation set [112]. Both sets contain the same main attack types, but differ in attack scenarios within the types. We refer to this data as NSLKDDfix.

We were also interested in the workings of the considered methods when the training and evaluation set have (approximately) the same underlying distribution. Therefore, we combined the training and evaluation set of NSLKDDfix into one set. For each experiment, this set was randomly split in a training and evaluation set. We refer to this data as NSLKDDrand.

UNSW-NB15

The UNSW-NB15 data was designed in 2015 by Moustafa and Slay. This dataset is much more recent than NSL-KDD and has more realistic aspects. In fact, it was constructed to address some of the inherent problems of NSL-KDD (for more information, see [78, 79]). We refer to this data as UNSWrand.

Moustafa and Slay also provide a fixed training and evaluation set that are statistically similar (in contrast to the provided training and evaluation set of NSL-KDD). We refer to this data as UNSWfix.

Preprocessing

Firstly, since the methods require numerical input, the categorical features in all four data settings were removed. In NSL-KDD these are Protocol_type, Service, Flag, and Difficulty_level. In UNSW-NB15 these variables are proto, state, service, stcpb, and dtcpb. Additionally, the features srcip, sport, dstip, dsport, Stime, and Ltime were removed, because they directly determine the output label or have no predictive use. Secondly, the response value was made binary: the benign class was made '0' and the malicious class '1'. Thirdly, all four data settings were standardised and Principal Component Analysis was performed to obtain (linearly) uncorrelated features and to reduce dimensionality [87]. To explain 99% of the variance, 31 Principal Components are necessary for NSLKDDfix and NSLKDDrand, 27 for UNSWrand, and 28 for UNSWfix.

7.5.2 Experiments

AL and ASeSL Methods

We considered four main AL and ASeSL techniques: *(i)* Plusmine (plu), *(ii)* a technique (tom) based on the work of Tomanek and Hahn [115], *(iii)* Plusmine-incomplete (pin), and *(iv)* the method from Chapter 6 Jasmine (jas). The technique tom incorporates the query approach of uncertainty sampling and automatically labels highly confident samples. These simple strategies make it a good benchmark. Plusmine-incomplete is Plusmine without the SeSL component (only AL). The difference in performance between plu and pin exactly shows the influence of Automatic Classification. Lastly, the differences in results between pin and jas demonstrate whether the changes made over Jasmine were indeed beneficial.

The intrusion detector is a supervised classifier and was one of two options in each method: Decision Tree (DeT) or Gradient Boosting Machine (GBM). Both learners are tree-based algorithms. DeT is the most basic one, since it consists of only one tree, while GBM integrates multiple boosted decision trees [30]. Hence, we incorporated a fast, but simple and weak predictor; or a slower, but more complex and highly flexible one. Methods that are paired with DeT obtain the suffix '.det' and the ones paired with GBM the suffix '.gbm', e.g., Plusmine with GBM is 'plu.gbm'.

Moreover, the anomaly detector was chosen to be Naive Bayes Classifier (NBC). This is a fast algorithm with no hyperparameters, making the results more robust. The technique is considered in all method settings, except for jas.gbm, since this is the original Jasmine procedure, which uses Isolation Forest (see Section 6.4.1).

Global Parameters

The global parameters were chosen before the experiments took place. These are the initial size of the labelled set $L(0)$, the initial size of the unlabelled pool $U(0)$, the size of the evaluation set E, the query set size Q, the maximum number of query observations N that are presented to the oracle (labelling budget), and specifically for Plusmine the number of subsets S that are available for AC (see Section 7.4.2). How many query iterations T were performed is given by: $T := \lfloor \min\{U(0), N\}/Q \rfloor$. Because of stochasticity, each experiment is repeated R times with $\mathcal{L}(0)$ and $\mathcal{U}(0)$ randomly drawn each repetition. The evaluation set $\mathcal{E}$ is also newly constructed for NSLKDDrand and UNSWrand. Hence, the average behaviour of the methods can be examined and the range in which the performance resides.

Firstly, we chose $L(0) = 200$, since a small starting size is usually the case in AL. Secondly, $E = 5{,}000$ was selected for NSLKDDrand and UNSWrand, as we deemed it large enough to be representative. The size was already provided for NSLKDDfix and UNSWfix by their corresponding authors with $E = 22{,}544$ and $E = 82{,}332$, respectively. Thirdly, $U(0)$ follows directly from $L(0)$ and E. Therefore, we had $U(0) = 143{,}317$ for NSLKDDrand, $U(0) = 125{,}773$ for NSLKDDfix, $U(0) = 2{,}534{,}847$ for UNSWrand, and $U(0) = 175{,}141$ for UNSWfix. Fourthly, we selected $Q = 50$, as we see adding 50 new observations sufficient for retraining the classifier. Finally, for Plusmine, $S = 50$ was chosen as a trade-off between computation time and a larger potential performance (see Section 7.4.2 for details).

Hyperparameter Tuning Gradient Boosting Machine

Table 7.1: Hyperparameter GBM ranges (sr = 'sample_rate')

distribution Bernoulli	histogram_type RoundRobin	learn_rate_annealing $\{0.95, 0.99, 0.999\}$
max_depth $\{6, 12, 24\}$	sr $\{0.60, 0.78, 1.0\}$	ntrees $\{250, 500, 1{,}000\}$
nbins $\{10, 16, 25\}$	nbins_cats $\{16, 32, 64\}$	learn_rate $\{0.02, 0.05, 0.125\}$
min_rows $\{6, 8, 10\}$	col_sr $\{0.84, 0.92, 1.0\}$	col_sr_change_per_level $\{0.94, 1.0, 1.06\}$
	col_sr_per_tree $\{0.40, 0.64, 1.0\}$	

GBM can be customised to a high degree, i.e., there are many hyperparameters. In each experiment, good values for these parameters were found by performing k-fold cross validation on $\mathcal{L}(0)$. The research of Tama and Rhee, the insights of Chapter 6, and exploratory studies were used to decide which hyperparameters were tuned and over which range [111]. Table 7.1 shows the selected hyperparameters and corresponding ranges for the `h2o.gbm` function of the `H2O.ai` package in the `R` programming language. The parameters that are not shown in the table got their default settings. Because the total number of combinations is enormous, random search was performed for 2 hours with the F_1 score as performance measure.

Hyperparameter Tuning AL and ASeSL

Table 7.2: Hyperparameter A(SeS)L ranges

$\alpha_a^{(0)}$	β, γ	τ	B
$\{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$	$\{\frac{1}{2}, 1, 2\}$	$\{\frac{1}{450}, \frac{1}{150}, \frac{1}{50}\}$	$\{25, 50, 100, 200\}$

The methods Plusmine, Plusmine-incomplete and Jasmine have AL-specific hyperparameters that had to be tuned. This tuning is part of the procedure. One of the hyperparameters was already introduced in Section 7.4.1: β, the weight factor in the information metrics. Additionally, we have $\alpha_a^{(0)}$, the initial anomaly query fraction; $\gamma > 0$, the update magnitude in α-dynamic updating; and $\tau > 0$, which is related to the query fraction of random observations. More information about these hyperparameters is provided in Section 6.4.8. The SeSL component of Plusmine introduces the hyperparameter B, the size of a candidate set for AC. Table 7.2 shows the ranges over which the hyperparameters were tuned. Now, let $\mathcal{H}$ be the set of all hyperparameter combinations. Thus, $|\mathcal{H}| = 81$ for Plusmine-incomplete and Jasmine, and $|\mathcal{H}| = 324$ for Plusmine.

During tuning, $\mathcal{L}(0)$ is randomly partitioned into the sets $\mathcal{L}_H(0)$, $\mathcal{U}_H(0)$, and $\mathcal{E}_H$ in a 30/50/20-split. As the notation suggests, $\mathcal{L}_H(0)$ is the initially labelled set for tuning with size $L_H(0)$, $\mathcal{U}_H(0)$ is the initially unlabelled set with size $U_H(0)$, and $\mathcal{E}_H$ is the evaluation set. In Plusmine and Plusmine-incomplete, the size of the query set Q_H during tuning is defined as $Q_H := \left\lceil \frac{L_H(0)}{L(0)} \cdot Q \right\rceil$. Thus, the ratio of Q_H to Q is the same as that of $L_H(0)$ to $L(0)$. The number of query iterations in tuning is equal to $t_H := \left\lceil \frac{U_H(0)}{Q_H} \right\rceil - 1$. The minus one is because the last iteration is not performed, since it always contains the leftover observations that are the least informative.

Next, let $h \in \mathcal{H}$ be some hyperparameter combination. Then, the chosen classifier (DeT or GBM) is trained on $\mathcal{L}_H(0)$ and applied to $\mathcal{E}_H$ to obtain the F_1 score $F_1^{(h)}(0)$. Then, the rest of the AL or ASeSL procedure is executed, as described in Sections 7.3 and 7.4. After this, the sequence of performance scores $(F_1^{(h)}(0), \dots, F_1^{(h)}(t_H))$ is obtained. Then, the area under the $(t, F_1^{(h)}(t))_{\{t=0,\dots,t_H\}}$-'curve' is determined to obtain a single quality score. Such a *learning curve* is commonly used in AL research to evaluate the overall performance of a prediction model [100]. The larger the area under the learning curve, the better combination $h \in \mathcal{H}$ is.

Because of stochasticity, the tuning rounds were repeated at least $R_H = 4$ times. Each repetition, $\mathcal{L}(0)$ was split into the three sets $\mathcal{L}_H(0)$, $\mathcal{U}_H(0)$, and $\mathcal{E}_H$, and the area under the learning curve was obtained. The combination that yielded the largest area averaged over the repetitions provided the (relatively) optimal values for $\alpha_a^{(0)}$, β, γ, τ, and (if applicable) B. In total, the DeT-based methods got 16 hours for this, while the GBM-based procedures got 14 hours, since they already used 2 hours for tuning the GBM classifier. If there was time left (usually for the lighter DeT-based methods), then another repetition was performed.

Assessment of AL and ASeSL Methods

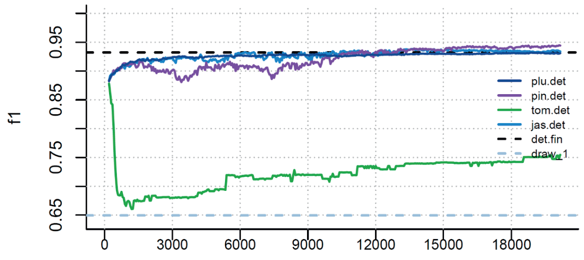
For the final assessment of the eight methods, the performance on the separate evaluation set $\mathcal{E}$ was used. In iteration t, the classifier was trained on $\mathcal{L}(t)$ and the performance score $F_1(t)$ was obtained for each method. After some reference iteration t_{ref} $(0 \leq t_{\text{ref}} \leq T)$, the area under the $(t, F_1(t))_{\{t=0,\dots,t_{\text{ref}}\}}$-curve $A(t_{\text{ref}})$ was calculated. Since the experiments were conducted R times, this yielded R reference areas per method. We used these areas to determine with a Mann-Whitney U test whether Plusmine performed significantly better.

7.6 Results

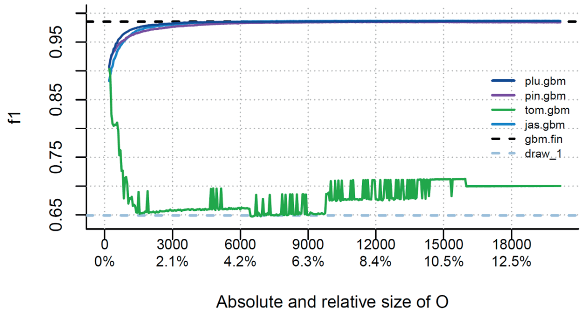
We discuss two categories of results: *(i)* average learning curves, and *(ii)* tables of p-values. The first category provides a graphical insight in how the performance of the four main methods (plu, pin, tom, and jas) evolved on the evaluation set $\mathcal{E}$. The second category describes the statistical significance of each method for several reference iterations by performing a Mann-Whitney U (MWU) test with significance level 0.05.

7.6.1 Results on NSL-KDD

NSLKDDrand



(a) DeT classifier



(b) GBM classifier

Figure 7.2: Learning curves on NSLKDDrand with random $\mathcal{E}$

Figure 7.2 shows the average learning curves of F_1 scores for the four main methods with DeT (Figure 7.2a) or GBM (Figure 7.2b) classifier. Each simulation yielded a learning curve and the figure shows the mean over all runs per method. The horizontal axis is the size of the *truly* labelled set $\mathcal{O}(t)$, i.e., the number of observations classified by the oracle. For both classifiers, there is a large discrepancy between tom and the other three methods. Moreover, the learning curves of plu, pin, and jas are very close to each other, especially for GBM. For the latter, the three methods quickly reached their final score, which is represented by the black dashed line (.fin). This is the average performance of the classifier trained on the complete training set with all labels available. As the initial performance was already good, there was not much to learn. It seems that Plusmine performed best for both DeT and GBM at the start, which is the most important part. It is also striking that Plusmine-incomplete performed less well than Jasmine. Furthermore, the difference between DeT or GBM is as expected: the F_1 scores for DeT were generally lower than those for GBM. The latter is a more complex technique and was therefore better able to learn structures in the data. Lastly, the grey dashed line is the Dutch Draw baseline whose framework was introduced in Chapter 5. This baseline is the maximum expected score that a classifier that makes random predictions or that predicts only one class can attain. The exact baseline for the F_1 score is presented in Table 5.4. Hence, the best non-learning strategy is to predict everything as malicious. An ML method should at least outperform this baseline.

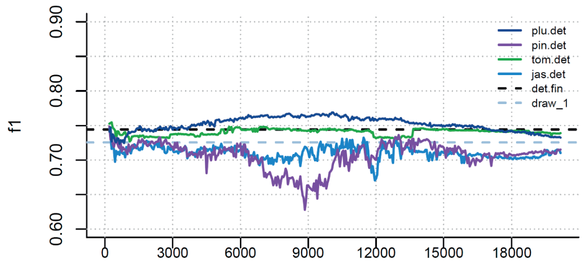
Table 7.3: p-values MWU test for NSLKDDrand

t_{ref}	$O(t_{\text{ref}})$	plu.d v. pin.d	plu.d v. tom.d	plu.d v. jas.d	pin.d v. jas.d
4	400	0.50	0.0054	0.75	0.78
10	700	0.52	$3.0 \cdot 10^{-12}$	0.32	0.66
25	1450	0.20	$7.9 \cdot 10^{-15}$	0.12	0.41
63	3350	0.00011	$7.9 \cdot 10^{-15}$	0.15	1.0
156	8000	$1.2 \cdot 10^{-10}$	$7.9 \cdot 10^{-15}$	0.20	1.0
399	20150	0.011	$7.9 \cdot 10^{-15}$	0.92	1.0
t_{ref}	$O(t_{\text{ref}})$	plu.g v. pin.g	plu.g v. tom.g	plu.g v. jas.g	pin.g v. jas.g
4	400	0.30	$7.4 \cdot 10^{-8}$	$1.3 \cdot 10^{-6}$	$4.3 \cdot 10^{-6}$
10	700	0.035	$2.0 \cdot 10^{-9}$	$3.0 \cdot 10^{-7}$	0.00013
25	1450	0.00052	$3.6 \cdot 10^{-11}$	$1.5 \cdot 10^{-7}$	0.046
63	3350	$2.1 \cdot 10^{-5}$	$5.3 \cdot 10^{-13}$	$1.3 \cdot 10^{-7}$	0.48
156	8000	$1.5 \cdot 10^{-5}$	$7.2 \cdot 10^{-12}$	$3.7 \cdot 10^{-5}$	0.92
399	20150	0.00010	$4.6 \cdot 10^{-11}$	0.039	1.0

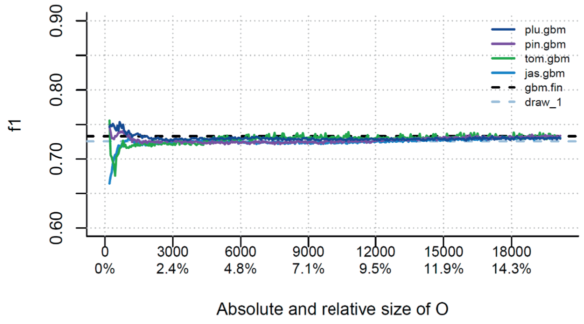
Table 7.3 supports what we observed in Figure 7.2. Note that '.d' represents '.det' and '.g' represents '.gbm'. A green value means that the first method in the corresponding column name performed significantly better than the second method, a red value means it was significantly worse, and a black value means no decisive conclusion could be drawn. Clearly, Plusmine performed significantly better than tom for all reference iterations. The bad performance of the latter was possibly due to the immediate automatic labelling of on average 80% of all unlabelled observations in the first few iterations. Especially for DeT, there were many mistakes in these predictions, making it very difficult for the method to recover from this. Also, plu.gbm was almost always significantly better than pin.gbm and jas.gbm. However, we see in Figure 7.2b that the differences in performance were small.

NSLKDDfix

Although NSLKDDfix is based on the same dataset as NSLKDDrand, the results are vastly different, as Figure 7.3 shows. Firstly, most surprisingly is the difference between DeT (Figure 7.3a) and GBM (Figure 7.3b). The final performance



(a) DeT classifier



(b) GBM classifier

Figure 7.3: Learning curves on NSLKDDfix – $\mathcal{E}$

of DeT was higher than that of GBM, even though GBM is a more complex technique. Also, plu.det performed better than plu.gbm after the first ten iterations. It even performed notably better than its final score. Secondly, this time tom was a worthy competitor. Interestingly, still approximately 80% of the observations were automatically labelled and the number of mistakes made was of the same order. Apparently, it has to do with the difference in distribution of training and evaluation set: tom was now better able to generalise, even though there were mistakes in the labelled set. Thirdly, pin was mostly better than or on par with jas, except for a large dip in pin.gbm around $\mathcal{O}(t) = 9{,}000$. Lastly, pin.det, jas.det, pin.gbm, tom.gbm, and jas.gbm performed worse or much worse than the Dutch Draw baseline at certain iterations. This is undesirable behaviour, because it means a better performance could be attained without learning and by predicting everything malicious, which the Dutch Draw classifier does in this specific setting. This means the benefits of AL or ASeSL are not clear here.

The p-values of the statistical tests in Table 7.4 suggest that Plusmine in general performed on par or significantly better than the other methods. There is one exception for $t_{\text{ref}} = 4$ and DeT.

Table 7.4: *p*-values MWU test for NSLKDDfix

t_{ref}	$O(t_{\text{ref}})$	plu.d v. pin.d	plu.d v. tom.d	plu.d v. jas.d	pin.d v. jas.d
4	400	0.12	0.98	0.11	0.37
10	700	0.16	0.89	0.0076	0.052
25	1450	0.0048	0.36	0.00048	0.16
63	3350	0.00052	0.050	$2.5 \cdot 10^{-5}$	0.12
156	8000	$1.1 \cdot 10^{-12}$	0.0042	$4.0 \cdot 10^{-12}$	0.48
399	20150	$7.9 \cdot 10^{-15}$	0.0068	$7.9 \cdot 10^{-15}$	0.74
t_{ref}	$O(t_{\text{ref}})$	plu.g v. pin.g	plu.g v. tom.g	plu.g v. jas.g	pin.g v. jas.g
4	400	0.071	0.031	$4.4 \cdot 10^{-7}$	$5.7 \cdot 10^{-7}$
10	700	0.14	0.0068	$4.4 \cdot 10^{-7}$	$8.3 \cdot 10^{-7}$
25	1450	0.24	0.0019	$5.3 \cdot 10^{-6}$	$7.1 \cdot 10^{-5}$
63	3350	0.088	0.0023	$5.9 \cdot 10^{-5}$	0.0051
156	8000	0.039	0.016	0.0011	0.088
399	20150	0.28	0.27	0.076	0.27

7.6.2 Results on UNSW-NB15

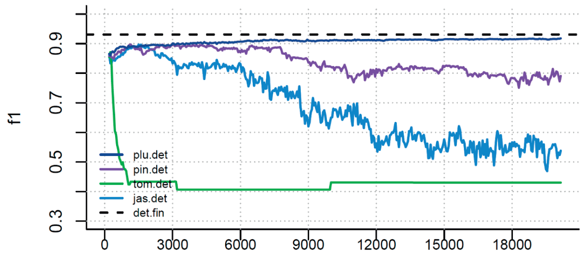
UNSWrand

Figure 7.4 shows the average learning curves for the UNSWrand setting. Remarkably, the results for GBM (Figure 7.4b) are very similar to the results on NSLKDDrand. Both datasets have in common that training and evaluation set have approximately the same distribution. However, the results with DeT (Figure 7.4a) are notably different for pin.det and jas.det. Both methods started comparably well to Plusmine, but after around 2,000 queried observations jas.det began to worsen, and the same happened for pin.det after approximately 7,500 queries.

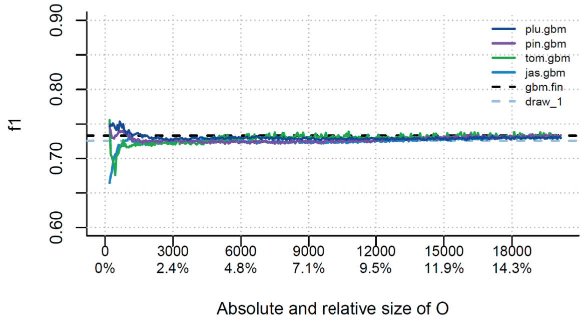
Table 7.5 confirms these observations as Plusmine was significantly better than the other methods for almost all reference values. In contrast to the results on NSL-KDD, it seems that pin is often better than jas, as indicated by the significant *p*-values in the final column of the tables and the learning curves.

UNSWfix

The learning curves on the UNSWfix data are shown in Figure 7.5. Again, the GBM classifier (Figure 7.5b) produced straightforward results: the learning curves of plu.gbm, pin.gbm, and jas.gbm were close to each other, and their initial performance was already close to the final score. This time, tom.gbm was very stable, but a bit lower than the other three. The results with DeT (Figure 7.5a) show more contrasting behaviours. plu.det and pin.det almost did not change and performed better than their final score, but tom.det and jas.det were much different. The first had a performance drop at the start,



(a) DeT classifier



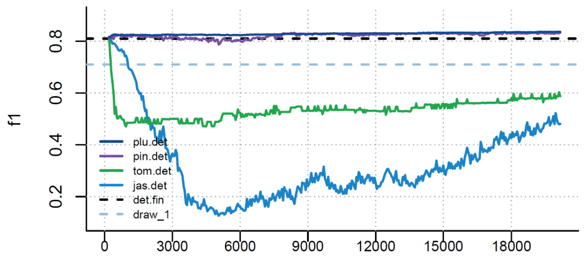
(b) GBM classifier

Figure 7.4: Learning curves on UNSWrand with random $\mathcal{E}$

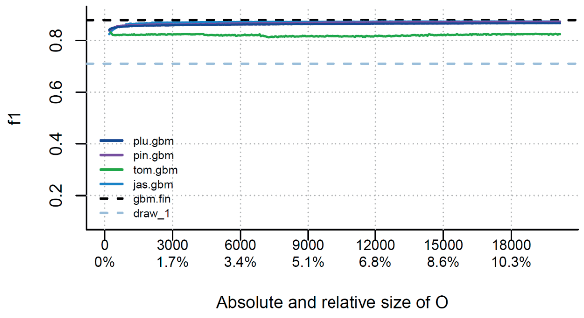
Table 7.5: p-values MWU test for UNSWrand

t_{ref}	$O(t_{\text{ref}})$	plu.d v. pin.d	plu.d v. tom.d	plu.d v. jas.d	pin.d v. jas.d
4	400	0.0040	0.022	0.0012	0.14
10	700	0.0033	$6.4 \cdot 10^{-8}$	$1.4 \cdot 10^{-5}$	0.030
25	1450	0.15	$3.5 \cdot 10^{-10}$	0.0057	0.040
63	3350	0.12	$3.0 \cdot 10^{-12}$	0.00032	0.0033
156	8000	0.011	$3.6 \cdot 10^{-13}$	$9.6 \cdot 10^{-12}$	$7.4 \cdot 10^{-8}$
399	20150	$1.2 \cdot 10^{-5}$	$3.2 \cdot 10^{-14}$	$7.9 \cdot 10^{-15}$	$9.4 \cdot 10^{-7}$
t_{ref}	$O(t_{\text{ref}})$	plu.g v. pin.g	plu.g v. tom.g	plu.g v. jas.g	pin.g v. jas.g
4	400	0.0020	$1.2 \cdot 10^{-5}$	$4.9 \cdot 10^{-9}$	0.00013
10	700	0.0024	$1.1 \cdot 10^{-8}$	$2.6 \cdot 10^{-8}$	0.00032
25	1450	0.0031	$1.5 \cdot 10^{-7}$	$8.3 \cdot 10^{-7}$	0.0064
63	3350	0.0038	$2.0 \cdot 10^{-7}$	0.00056	0.24
156	8000	0.0027	$8.3 \cdot 10^{-7}$	0.032	0.90
399	20150	0.0019	$1.1 \cdot 10^{-6}$	0.19	0.97

and then tried to recover. As before, most of the unlabelled observations were immediately automatically labelled. jas.det had a fast performance decrease



(a) DeT classifier



(b) GBM classifier

Figure 7.5: Learning curves on UNSWfix – $\mathcal{E}$

and later on tried to rebound. These results clearly show the benefit of pin over jas.

Table 7.6: p-values MWU test for UNSWfix

t_{ref}	$O(t_{\text{ref}})$	plu.d v. pin.d	plu.d v. tom.d	plu.d v. jas.d	pin.d v. jas.d
4	400	0.0057	$3.0 \cdot 10^{-7}$	$4.4 \cdot 10^{-7}$	0.00087
10	700	0.035	$2.4 \cdot 10^{-13}$	$1.2 \cdot 10^{-10}$	$4.1 \cdot 10^{-8}$
25	1450	0.019	$1.6 \cdot 10^{-14}$	$7.9 \cdot 10^{-15}$	$5.5 \cdot 10^{-14}$
63	3350	$7.7 \cdot 10^{-5}$	$3.2 \cdot 10^{-14}$	$7.9 \cdot 10^{-15}$	$7.9 \cdot 10^{-15}$
156	8000	$1.1 \cdot 10^{-5}$	$5.5 \cdot 10^{-14}$	$7.9 \cdot 10^{-15}$	$7.9 \cdot 10^{-15}$
399	20150	0.00060	$5.5 \cdot 10^{-14}$	$7.9 \cdot 10^{-15}$	$7.9 \cdot 10^{-15}$
t_{ref}	$O(t_{\text{ref}})$	plu.g v. pin.g	plu.g v. tom.g	plu.g v. jas.g	pin.g v. jas.g
4	400	0.25	0.0012	0.066	0.17
10	700	0.82	0.00022	0.55	0.34
25	1450	0.99	$2.1 \cdot 10^{-5}$	0.98	0.81
63	3350	1.0	$7.4 \cdot 10^{-6}$	1.0	0.98
156	8000	1.0	$3.8 \cdot 10^{-6}$	1.0	1.0
399	20150	1.0	$5.3 \cdot 10^{-6}$	1.0	0.99

Finally, the p-values in Table 7.6 support what the corresponding learning curves with DeT suggest: Plusmine was significantly better than the other methods for all reference values, and Plusmine-incomplete always performed better than jas.det. This is in contrast to the p-values for GBM: they seem to not be in favour of Plusmine at all. pin.gbm and jas.gbm firstly performed on par with plu.gbm, but were later on significantly better. Also, the final column demonstrates that Jasmine outperforms pin.gbm. However, Figure 7.5b shows how close the performance scores actually were to each other.

7.7 Discussion

In this chapter, we introduced Plusmine, an NIDS that combines new techniques for both AL and SeSL. Its AL component is an improved version of our Jasmine method introduced in Chapter 6. The results on the UNSW-NB15 data support the improvement in most cases; and when this was not the case, the difference between the learning curves was small. However, this was not the case for the NSL-KDD data. Yet, in general, our query approach in Plusmine performed more reliably than Jasmine, i.e., drops in performance were less severe.

Secondly, the SeSL component of Plusmine had a convincingly positive effect on performance for all dataset configurations. It performed at least on par with the other methods, but often it was significantly better. The rare occasion when this was not the case, the difference between the learning curves was small. Hence, the addition of AC is beneficial. Moreover, our proposed form of AC performed mostly significantly better than the benchmark ASeSL method tom that automatically labels confident observations, as is common practice for SeSL [115].

Although Plusmine obtains good, more robust and reliable results, there is room for improvement. Firstly, its query function does not consistently outperform the benchmark approaches. Hence, we suggest for future research to analyse what can be improved further in the dynamic query approach, also from a mathematical standpoint. Secondly, since the initial performance of the classifiers was close to the final score, there was not much room for learning. This makes comparing the methods sometimes challenging. Lastly, and this holds for most cyber security research, it is not clear how Plusmine behaves on actual computer network data. Therefore, we would like to apply it in such a setting.

8 Conclusion

Nou ja, ik heb lekker geluncht.

Georgina Verbaan
Wie is de Mol? 2008

In this chapter, we recap the topics that we discussed in this dissertation and give answers to the research questions (RQs) that were posed in Chapter 1. Based on the answers, we give an overall conclusion, reflect, and provide directions for further research.

We explored three research topics that all contribute to *molding the symbiosis between human and machine*:

- *Anomaly Detection* (AD)
- *Model Evaluation* (ME)
- *Active Learning* (AL).

8.1 Anomaly Detection

In Part I, we considered two case studies of AD with raw datasets that were labelled to different degrees.

RQ 1a: *How can we make the raw data of two real case studies usable for Anomaly Detection?*

In Chapter 2, we obtained a partially labelled data of a realistic international cyber defence case study. First, we cleaned up the data by discarding non-informative features and a small fraction of observations that had missing values. Then, we engineered and extracted new variables using the raw data and domain knowledge. These procedures showed to be essential in creating a usable dataset for AD: four out of the five most influential features for detection were engineered by us.

The case study in Chapter 3 consisted of aeroplane booking data from online travel agencies. The multiple datasets were combined into one usable set by using domain knowledge and feature engineering. The new features demonstrated to be crucial for detection: novel fraudulent bookings were discovered because of them. Moreover, we demonstrated the effects on fraud detection when the data was either normalised or standardised. It seemed that these transformations were not generally beneficial as some AD techniques performed less well. This means one should carefully consider whether or not to normalise or standardise data given the chosen detection method.

RQ 1*b*: *What are the strengths and weaknesses of the Supervised Learning and Unsupervised Learning techniques used in these case studies?*

In Chapter 2, we used the Autoencoder method for Unsupervised Learning (UL) and the Gradient Boosting Machine (GBM) for Supervised Learning (SL). The Autoencoder does not work with labels, but we used the known malicious observations to tune the hyperparameters and to evaluate the results. For the GBM, we assumed the unlabelled observations to be benign during training and tuning. In Chapter 3, we exclusively used UL methods. We found substantial differences between UL and SL in the two case studies. The unsupervised Machine Learning (ML) techniques were good at detecting unknown malicious instances, as the Autoencoder found many new cyber attacks and the UL methods applied to online booking data discovered novel fraudulent bookings. Additionally, the human experts ensured that they would not have found these malicious instances by themselves, which really shows the importance of using ML in these domains. However, the unsupervised techniques performed less well in retrieving known malicious data: the Autoencoder did not assign the highest anomaly scores to the known intrusions. This is in contrast to SL, as the GBM achieved a high detection rate of known cyber attacks. These outcomes clearly demonstrate the strengths and weaknesses of SL and UL under these assumptions, and suggest that combining the two paradigms could improve performance, such as in Semi-Supervised Learning (SeSL).

RQ 1*c*: *What are the benefits of using a human domain expert to inspect the anomalous instances that the Machine Learning technique discovered?*

Human domain experts were involved in both case studies. In Chapter 2, the cyber analyst investigated two sets of observations: one with highly anomalous connections, and one with normal observations. Taking samples from sets with different characteristics allowed us to fairly assess the detection performances of the AD techniques. We found that many more new cyber attacks were discovered in the anomalous set compared to the normal set. In Chapter 3, a similar procedure was performed. Per AD technique, three sets were constructed: one with highly anomalous bookings, one with normal instances, and one set with anomaly scores in between. Again, new fraudulent instances were mostly found in the set with highly anomalous bookings. Hence, by letting domain experts label specifically selected samples, malicious data could be found more effectively and labelling was done more efficiently. The inclusion of a human expert in these studies can therefore be seen as a stepping stone towards AL.

8.2 Model Evaluation

In Part II, we took a fundamental approach to ME: we constructed a new evaluation metric and created universal baselines.

RQ 2*a*: *Is it possible to construct an accurate and robust equivalent to the common F_1 score for partially labelled data?*

We showed in Chapter 4 that it is possible to create an accurate and robust equivalent to the F_1 score for partially labelled data. We explored the field of Positive-Unlabelled Learning in which the labelled instances are only positive. We made the realistic assumption that the labelled observations were selected completely at random, i.e., each actual positive instance had the same probability ρ of being labelled. This allowed us to construct $\overline{F_1}$, our approximation of the real F_1 score. First, we mathematically analysed how sensitive $\overline{F_1}$ is towards mistakes in the estimation of ρ. Second, we experimentally demonstrated that $\overline{F_1}$ is better than the well-known LL score (see [55]) in two ways: *(i)* $\overline{F_1}$ is significantly closer to the true F_1 score, and *(ii)* $\overline{F_1}$ more often selects the real optimal model out of a set of models (with the optimal model achieving the highest real F_1 score).

RQ 2*b*: *Is there a general, simple, but informative way to benchmark any evaluation metric for binary classification? In other words, can a metric-specific baseline be constructed that any newly developed Machine Learning method should clearly outperform?*

In Chapter 5, we examined whether we could construct a framework that is able to generate a *general*, *simple*, and *informative* baseline for binary model performance. We developed the Dutch Draw methodology, which assumes that the four base measures (TP, FN, TN, and FP) have specific hypergeometric distributions. These distributions do not use information from the feature values in the dataset. By making use of probability theory, we derived baselines for many evaluation metrics. Such a Dutch Draw baseline is *general*, because it can be applied to any binary prediction problem. The baseline is *simple*, because it is derived from elementary (stochastic) rules, and is therefore easy to calculate, and it should be outperformed by any ML model. Also, our baseline is *informative*, because it is by definition derived from the optimal distribution. These three properties ensure that the Dutch Draw baseline is a valuable addition to the evaluation of any ML model.

8.3 Active Learning

In Part III, we dove into the field of AL and developed methodologies that make human labelling of data more efficient.

RQ 3*a*: *What are the effects on the prediction performance and the labelling process when the query function can dynamically adjust itself to best fit the current situation?*

The focus of Chapter 6 was on the development of a dynamic AL methodology for network intrusion detection. Most query functions are static: their selection rule is determined at the start of the labelling process and does not change when more labels become available. We moved away from this concept and proposed our *Jasmine* method with dynamic updating. Each labelling iteration, the query set is composed of anomalous, uncertain, and randomly selected observations. The crucial part is that the current state of Jasmine directly determines the number of instances of each type that will be chosen. If the method notices that, for example, uncertain observations increase prediction performance more than anomalous instances, then it updates the query fractions such that more uncertain observations are queried next iteration. We saw that Jasmine was indeed better able to adapt to the labelled dataset than methods without dynamic updating, since it achieved good and robust results over multiple datasets while the benchmark methods had fluctuating outcomes.

RQ 3*b*: *What are the effects on the prediction performance when Semi-Supervised Learning is used besides Active Learning to automatically label observations?*

AL can greatly increase labelling efficiency. However, it is limited by the labelling speed of the oracle. Therefore, in Chapter 7, we examined the inclusion of SeSL as an additional labelling source. We proposed *Plusmine*, our Active SeSL (ASeSL) method that consists of new techniques for both its AL and SeSL components. The latter component constitutes Automatic Classification (AC). Each labelling iteration, Plusmine takes multiple random subsets from the unlabelled data and virtually adds each subset with their predicted labels to the current labelled pool. A classifier is trained on each expanded labelled set and its prediction performance is assessed. The subset that constitutes the best result is then officially automatically labelled and removed from the unlabelled pool. We showed that this new source of labelling had a favourable effect on prediction performance compared to the benchmark ASeSL method and to AL approaches without AC.

8.4 Concluding Remarks and Outlook

In this dissertation, we showed how to mold the symbiosis between human and machine in multiple ways. First, we demonstrated the benefits of including a human expert in AD as new malicious instances were found by combining ML and domain knowledge. Second, we improved human understanding of model prediction performance by constructing a new evaluation metric and by designing a framework to benchmark performance scores. Third, we have taken important steps in AL, the field in which human and machine collaborate to increase labelling efficiency. In this section, we address a number of challenging topics for further research.

Anomaly Detection

First of all, we considered only two specific case studies for AD. The transformation of raw data into usable datasets was based on specific domain knowledge. We see this as a strength of our research, because the newly constructed features were deemed to be important in detection. However, it is difficult to generalise the procedure of feature engineering. Second, we could have extended the evaluation procedure. As an example, the samples that were selected for the human expert in Chapter 2 were based on the results of the Autoencoder. A next step is to apply the same procedure for the GBM. Lastly, since we had partially labelled data in the first case study, an interesting question is what the detection performance of SeSL methods would be.

Model Evaluation

Our estimator $\overline{F_1}$ is unbiased when the probability ρ of a positive instance being labelled is fixed, but it is expected to overestimate the real F_1 score when ρ is noisy. The estimation can exceed 1, even though $F_1 \leq 1$. The likelihood of this overestimation occurring should be analysed. More generally speaking, our research opens the door to constructing more equivalents to evaluation metrics for partially labelled data.

For the Dutch Draw, it is clear that an ML model should outperform our baseline, but we cannot quantify how much better the model is if it outperforms the baseline. Explicitly deriving how much information was learned compared to the baseline is highly valuable. Also, the Dutch Draw framework should be extended such that it can benchmark ML methods for multi-class and continuous prediction problems too.

Active Learning

The α-dynamic updating approach in Jasmine and Plusmine has shown that the relevance of querying certain types of observations can change during the AL process. More research should be executed on this concept, since we proposed one dynamic framework, but there are many more possibilities. Reinforcement

Learning (RL) could be incorporated in AL as by definition RL makes decisions in a dynamic environment in order to achieve a certain goal. Another opportunity is to consider human uncertainty in labelling. It is assumed that the oracle is omniscient, and thus, makes no mistakes, which is possibly not realistic. Incorporating uncertainty in querying greatly expands the field of AL and its applicability in practice.

Publications

[45] J. G. Klein, S. Bhulai, M. Hoogendoorn and R. D. van der Mei. ‘Jasmine: A new Active Learning approach to combat cybercrime’. In: *Machine Learning with Applications* 9 (2022), page 100351.

[46] J. G. Klein, S. Bhulai, M. Hoogendoorn and R. D. van der Mei. ‘Plusmine: Dynamic Active Learning with Semi-Supervised Learning for Automatic Classification’. In: *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence*. IEEE. 2021.

[47] J. G. Klein, S. Bhulai, M. Hoogendoorn, R. D. van der Mei and R. Hinfelaar. ‘Detecting Network Intrusion Beyond 1999: Applying Machine Learning Techniques to a Partially Labeled Cybersecurity Dataset’. In: *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence*. IEEE. 2018, pages 784–787.

[72] C. Mensah, J. G. Klein, S. Bhulai, M. Hoogendoorn and R. D. van der Mei. ‘Detecting Fraudulent Bookings of Online Travel Agencies with Unsupervised Machine Learning’. In: *Proceedings of International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer. 2019, pages 334–346.

[109] S. A. Tabatabaei, J. G. Klein and M. Hoogendoorn. ‘Estimating the F_1 Score for Learning from Positive and Unlabeled Examples’. In: *Proceedings of International Conference on Machine Learning, Optimization, and Data Science*. Springer. 2020, pages 150–161.

[118] E. P. van de Bijl, J. G. Klein, J. Pries, S. Bhulai, M. Hoogendoorn and R. D. van der Mei. ‘The Dutch Draw: Constructing a Universal Baseline for Binary Prediction Models’. In: (Submitted for publication in 2022).

Bibliography

[2] M. Ahmed, A. N. Mahmood and J. Hu. ‘A Survey of Network Anomaly Detection Techniques’. In: *Journal of Network and Computer Applications* 60 (2016), pages 19–31.

[3] M. Almgren and E. Jonsson. ‘Using Active Learning in Intrusion Detection’. In: *Proceedings of 17th IEEE Computer Security Foundations Workshop*. IEEE. 2004, pages 88–98.

[4] R. d. A. Araújo, A. L. Oliveira and S. Meira. ‘A Morphological Neural Network for Binary Classification Problems’. In: *Engineering Applications of Artificial Intelligence* 65 (2017), pages 12–28.

[5] J. Balayla. ‘Prevalence Threshold (ϕ e) and the Geometry of Screening Curves’. In: *PLoS One* 15.10 (2020), e0240215.

[6] J. Bekker and J. Davis. ‘Learning from Positive and Unlabeled Data: A Survey’. In: *Machine Learning* 109.4 (2020), pages 719–760.

[7] C. M. Bishop. ‘Pattern Recognition and Machine Learning’. In: *Machine Learning* 128.9 (2006).

[8] R. J. Bolton and D. J. Hand. ‘Unsupervised Profiling Methods for Fraud Detection’. In: *Credit Scoring and Credit Control VII* (2001), pages 235–255.

[9] S. Budd, E. C. Robinson and B. Kainz. ‘A Survey on Active Learning and Human-in-the-Loop Deep Learning for Medical Image Analysis’. In: *Medical Image Analysis* (2021), page 102062.

[10] L. M. Candanedo and V. Feldheim. ‘Accurate Occupancy Detection of an Office Room from Light, Temperature, Humidity and CO2 Measurements using Statistical Learning Models’. In: *Energy and Buildings* 112 (2016), pages 28–39.

[11] R. Caruana, N. Karampatziakis and A. Yessenalina. 'An Empirical Evaluation of Supervised Learning in High Dimensions'. In: *Proceedings of the 25th International Conference on Machine Learning*. 2008, pages 96–103.

[13] R. Chalapathy and S. Chawla. 'Deep Learning for Anomaly Detection: A Survey'. In: *arXiv:1901.03407* (2019).

[14] V. Chandola, A. Banerjee and V. Kumar. 'Anomaly Detection: A Survey'. In: *ACM Computing Surveys (CSUR)* 41.3 (2009), pages 1–58.

[15] O. Chapelle, B. Schölkopf and A. Zien. 'Semi-Supervised Learning'. In: *IEEE Transactions on Neural Networks* 20.3 (2009), pages 542–542.

[16] N. Chinchor. 'MUC-4 Evaluation Metrics'. In: *Proceedings of the 4th Conference on Message Understanding*. Association for Computational Linguistics, 1992, pages 22–29.

[17] N. Chinchor and B. M. Sundheim. 'MUC-5 Evaluation Metrics'. In: *Proceedings of the 5th Conference on Message Understanding*. 1993, pages 69–78.

[18] M. Claesen and B. De Moor. 'Hyperparameter Search in Machine Learning'. In: *arXiv:1502.02127* (2015).

[20] R. Couronné, P. Probst and A.-L. Boulesteix. 'Random Forest Versus Logistic Regression: a Large-scale Benchmark Experiment'. In: *BMC Bioinformatics* 19.1 (2018), pages 1–14.

[21] A. P. Dempster, N. M. Laird and D. B. Rubin. 'Maximum Likelihood from Incomplete Data via the EM Algorithm'. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pages 1–22.

[22] F. Denis, R. Gilleron and M. Tommasi. 'Text Classification from Positive and Unlabeled Examples'. In: *Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. 2002, pages 1927–1934.

[23] L Dhanabal and S. Shantharajah. 'A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms'. In: *International Journal of Advanced Research in Computer and Communication Engineering* 4.6 (2015), pages 446–452.

[25] M. Elahi, F. Ricci and N. Rubens. 'A Survey of Active Learning in Collaborative Filtering Recommender Systems'. In: *Computer Science Review* 20 (2016), pages 29–50.

[26] C. Elkan and K. Noto. 'Learning Classifiers from Only Positive and Unlabeled Data'. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2008, pages 213–220.

[27] Z. Ferdousi and A. Maeda. 'Unsupervised Outlier Detection in Time Series Data'. In: *Proceedings of the 22nd International Conference on Data Engineering Workshops*. IEEE. 2006, pages 51–56.

[28] M. A. Ferrag, L. Maglaras, S. Moschoyiannis and H. Janicke. ‘Deep Learning for Cyber Security Intrusion Detection: Approaches, Datasets, and Comparative Study’. In: *Journal of Information Security and Applications* 50 (2020), page 102419.

[29] E. B. Fowlkes and C. L. Mallows. ‘A Method for Comparing Two Hierarchical Clusterings’. In: *Journal of the American Statistical Association* 78.383 (1983), pages 553–569.

[30] J. H. Friedman. ‘Greedy Function Approximation: a Gradient Boosting Machine’. In: *Annals of Statistics* 9.5 (2001), pages 1189–1232.

[31] A. Gadde, E. E. Gad, S. Avestimehr and A. Ortega. ‘Active Learning for Community Detection in Stochastic Block Models’. In: *Proceedings of the IEEE International Symposium on Information Theory*. IEEE. 2016, pages 1889–1893.

[33] P. Gogoi, B. Borah and D. K. Bhattacharyya. ‘Anomaly Detection Analysis of Intrusion Data Using Supervised & Unsupervised Approach’. In: *Journal of Convergence Information Technology* 5.1 (2010), pages 95–110.

[34] M. Goldstein and S. Uchida. ‘A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data’. In: *PloS One* 11.4 (2016), e0152173.

[35] P. Gonçalves, M. Araújo, F. Benevenuto and M. Cha. ‘Comparing and Combining Sentiment Analysis Methods’. In: *Proceedings of the 1st ACM Conference on Online Social Networks*. 2013, pages 27–38.

[36] N. Görnitz, M. Kloft, K. Rieck and U. Brefeld. ‘Active Learning for Network Intrusion Detection’. In: *Proceedings of the 2nd ACM Workshop on Security and Artificial Intelligence*. 2009, pages 47–54.

[37] N. Görnitz, M. Kloft, K. Rieck and U. Brefeld. ‘Toward Supervised Anomaly Detection’. In: *Journal of Artificial Intelligence Research* 46 (2013), pages 235–262.

[38] Y. Gu and D. Zydek. ‘Active Learning for Intrusion Detection’. In: *Proceedings of the National Wireless Research Collaboration Symposium*. IEEE. 2014, pages 117–122.

[39] J. L. Guerra Torres, C. A. Catania and E. Veas. ‘Active Learning Approach to Label Network Traffic Datasets’. In: *Journal of Information Security and Applications* 49 (2019), page 102388.

[40] M. F. A. Hady and F. Schwenker. ‘Combining Committee-based Semi-supervised Learning and Active Learning’. In: *Journal of Computer Science and Technology* 25.4 (2010), pages 681–698.

[41] F. E. Heba, A. Darwish, A. E. Hassanien and A. Abraham. ‘Principle Components Analysis and Support Vector Machine Based Intrusion Detection System’. In: *Proceedings of the 10th International Conference on Intelligent Systems Design and Applications*. IEEE. 2010, pages 363–367.

[42] K. Heller, K. Svore, A. D. Keromytis and S. Stolfo. 'One Class Support Vector Machines for Detecting Anomalous Windows Registry Accesses'. In: *Proceedings of the Workshop on Data Mining for Computer Security.* IEEE. 2003.

[43] L. M. Ibrahim, D. T. Basheer and M. S. Mahmod. 'A Comparison Study for Intrusion Database (KDD99, NSL-KDD) Based on Self Organization Map (SOM) Artificial Neural Network'. In: *Journal of Engineering Science and Technology* 8.1 (2013), pages 107–119.

[44] M. I. Jordan and T. M. Mitchell. 'Machine Learning: Trends, Perspectives, and Prospects'. In: *Science* 349.6245 (2015), pages 255–260.

[48] J. Kleinberg. 'An Impossibility Theorem for Clustering'. In: *Advances in Neural Information Processing Systems* (2003), pages 463–470.

[49] O. O. Koyejo, N. Natarajan, P. K. Ravikumar and I. S. Dhillon. 'Consistent Binary Classification with Generalized Performance Metrics'. In: *Advances in Neural Information Processing Systems* 27 (2014).

[50] M. Kubat, R. C. Holte and S. Matwin. 'Machine Learning for the Detection of Oil Spills in Satellite Radar Images'. In: *Machine Learning* 30.2 (1998), pages 195–215.

[51] B. Kulich. 'Lessons Learned from Military Cyber Defence Exercises'. In: *Science & Military Journal* 9.1 (2014), page 47.

[52] P. Kumar and A. Gupta. 'Active Learning Query Strategies for Classification, Regression, and Clustering: a Survey'. In: *Journal of Computer Science and Technology* 35.4 (2020), pages 913–945.

[53] T. O. Kvålseth. 'Note on Cohen's Kappa'. In: *Psychological Reports* 65.1 (1989), pages 223–226.

[54] I. Lee. 'Big Data: Dimensions, Evolution, Impacts, and Challenges'. In: *Business Horizons* 60.3 (2017), pages 293–303.

[55] W. S. Lee and B. Liu. 'Learning with Positive and Unlabeled Examples Using Weighted Logistic Regression'. In: *Proceedings of the International Conference on Machine Learning.* Volume 3. 2003, pages 448–455.

[56] Y. Leng, X. Xu and G. Qi. 'Combining Active Learning and Semi-supervised Learning to Construct SVM Classifier'. In: *Knowledge-Based Systems* 44 (2013), pages 121–131.

[57] J. Levatić, M. Ceci, D. Kocev and S. Džeroski. 'Semi-Supervised Classification Trees'. In: *Journal of Intelligent Information Systems* 49.3 (2017), pages 461–486.

[58] D. D. Lewis and W. A. Gale. 'A Sequential Algorithm for Training Text Classifiers'. In: *Proceedings of the Special Interest Group on Information Retrieval.* Springer. 1994, pages 3–12.

[59] X. Li and B. Liu. 'Learning to Classify Texts Using Positive and Unlabeled Data'. In: *Proceedings of the 18th International Joint Conference on Artificial Intelligence.* 2003, pages 587–592.

[60] Y. Li and L. Guo. 'An Active Learning Based TCM-KNN Algorithm for Supervised Network Intrusion Detection'. In: *Computers & Security* 26.7-8 (2007), pages 459–467.

[61] Z. C. Lipton, C. Elkan and B. Naryanaswamy. 'Optimal Thresholding of Classifiers to Maximize F1 Measure'. In: *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2014, pages 225–239.

[62] B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer Science & Business Media, 2007.

[63] B. Liu, W. S. Lee, P. S. Yu and X. Li. 'Partially Supervised Classification of Text Documents'. In: *Proceedings of the 19th International Conference on Machine Learning*. 2002, pages 387–394.

[64] F. T. Liu, K. M. Ting and Z.-H. Zhou. 'Isolation-based Anomaly Detection'. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6.1 (2012), pages 1–39.

[65] F. T. Liu, K. M. Ting and Z.-H. Zhou. 'Isolation Forest'. In: *Proceedings of the 8th IEEE International Conference on Data Mining*. IEEE. 2008, pages 413–422.

[66] S. Lloyd. 'Least Squares Quantization in PCM'. In: *IEEE Transactions on Information Theory* 28.2 (1982), pages 129–137.

[67] K. Maennel, R. Ottis and O. Maennel. 'Improving and Measuring Learning Effectiveness at Cyber Defense Exercises'. In: *Proceedings of the Nordic Conference on Secure IT Systems*. Springer. 2017, pages 123–138.

[68] M. Małowidzki, P Berezinski and M. Mazur. 'Network Intrusion Detection: Half a Kingdom for a Good Dataset'. In: *Proceedings of NATO STO SAS-139 Workshop, Portugal*. 2015.

[69] C.-H. Mao, H.-M. Lee, D. Parikh, T. Chen and S.-Y. Huang. 'Semi-supervised Co-training and Active Learning Based Approach for Multi-view Intrusion Detection'. In: *Proceedings of the ACM Symposium on Applied Computing*. 2009, pages 2042–2048.

[70] B. W. Matthews. 'Comparison of the Predicted and Observed Secondary Structure of T4 Phage Lysozyme'. In: *Biochimica et Biophysica Acta (BBA)-Protein Structure* 405.2 (1975), pages 442–451.

[71] Y. Meng and L.-f. Kwok. 'Intrusion Detection Using Disagreement-Based Semi-Supervised Learning: Detection Enhancement and False Alarm Reduction'. In: *International Symposium on Cyberspace Safety and Security*. Springer. 2012, pages 483–497.

[73] J. H. Min and C. Jeong. 'A Binary Classification Method For Bankruptcy Prediction'. In: *Expert Systems with Applications* 36.3 (2009), pages 5256–5263.

[74] T. Mitchell. *Machine Learning*. New York: McGraw-Hill, Inc, 1997.

[76] S. Moro, P. Cortez and P. Rita. ‘A Data-driven Approach to Predict the Success of Bank Telemarketing’. In: *Decision Support Systems* 62 (2014), pages 22–31.

[77] S. A. Mouloua, J. Ferraro, M. Mouloua, G. Matthews and R. R. Copeland. ‘Trend Analysis of Cyber Security Research Published in HFES Proceedings from 1980 to 2018’. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Volume 63. 1. SAGE Publications Sage CA: Los Angeles, CA. 2019, pages 1600–1604.

[78] N. Moustafa and J. Slay. ‘The Evaluation of Network Anomaly Detection Systems: Statistical Analysis of the UNSW-NB15 Data Set and the Comparison with the KDD99 Data Set’. In: *Information Security Journal: A Global Perspective* 25.1-3 (2016), pages 18–31.

[79] N. Moustafa and J. Slay. ‘UNSW-NB15: a Comprehensive Data Set for Network Intrusion Detection Systems’. In: *Proceedings of the Military Communications and Information Systems Conference*. IEEE. 2015, pages 1–6.

[80] G. Muhammad and M. Melhem. ‘Pathological Voice Detection and Binary Classification Using MPEG-7 Audio Features’. In: *Biomedical Signal Processing and Control* 11 (2014), pages 1–9.

[82] A. Natekin and A. Knoll. ‘Gradient Boosting Machines, a Tutorial’. In: *Frontiers in Neurorobotics* 7 (2013), page 21.

[84] J. O. Ogutu, H.-P. Piepho and T. Schulz-Streeck. ‘A Comparison of Random Forests, Boosting and Support Vector Machines for Genomic Selection’. In: *BMC Proceedings*. Volume 5. BioMed Central. 2011, pages 1–5.

[85] W. Palmer and R. Allen. ‘Note on the Accuracy of Forecasts Concerning the Rain Problem’. In: *US Weather Bureau* 4 (1949).

[87] K. Pearson. ‘LIII. On Lines and Planes of Closest Fit to Systems of Points in Space’. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pages 559–572.

[88] D. Pelleg and A. Moore. ‘Active Learning for Anomaly and Rare-category Detection’. In: *Advances in Neural Information Processing Systems* 17 (2004), pages 1073–1080.

[89] Z. Qiu, D. J. Miller and G. Kesidis. ‘Flow Based Botnet Detection Through Semi-supervised Active Learning’. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2017, pages 2387–2391.

[90] H. Raeisi Shahraki, S. Pourahmad and N. Zare. ‘Important Neighbors: A Novel Approach to Binary Classification in High Dimensional Data’. In: *BioMed Research International* (2017).

[91] M. Reif, M. Goldstein, A. Stahl and T. M. Breuel. ‘Anomaly Detection by Combining Decision Trees and Parametric Densities’. In: *Proceedings of the 19th International Conference on Pattern Recognition*. IEEE. 2008, pages 1–4.

[92] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen and X. Wang. ‘A Survey of Deep Active Learning’. In: *ACM Computing Surveys* 54.9 (2021), pages 1–40.

[93] S Revathi and A Malathi. ‘A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection’. In: *International Journal of Engineering Research & Technology* 2.12 (2013), pages 1848–1853.

[96] J. T. Schaefer. ‘The Critical Success Index as an Indicator of Warning Skill’. In: *Weather and Forecasting* 5.4 (1990), pages 570–575.

[97] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor and J. C. Platt. ‘Support Vector Method for Novely Detection’. In: *Advances in Neural Information Processing Systems*. Volume 12. 1999, pages 582–588.

[98] F. Schuetz and S. Burschka. *Locked Shields: NATO Cyber Defense Exercise 2012. Exercise Report*. Technical report. RUAG Holding AG Bern, Switzerland, 2012.

[99] G. Sergioli, R. Giuntini and H. Freytes. ‘A New Quantum Approach to Binary Classification’. In: *PloS One* 14.5 (2019), e0216224.

[100] B. Settles. *Active Learning Literature Survey*. Technical report. University of Wisconsin-Madison Department of Computer Sciences, 2009.

[101] P. W. Singer and A. Friedman. *Cybersecurity: What Everyone Needs to Know*. OUP USA, 2014.

[102] M. Skala. ‘Hypergeometric Tail Inequalities: Ending The Insanity’. In: *arXiv:1311.5939* (2013).

[103] R. Sommer and V. Paxson. ‘Outside The Closed World: On Using Machine Learning for Network Intrusion Detection’. In: *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE. 2010, pages 305–316.

[104] J. W. Stokes, J. Platt, J. Kravis and M. Shilman. *ALADIN: Active Learning of Anomalies to Detect Intrusions*. Technical report. Microsoft Research, 2008.

[105] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis and P. K. Chan. ‘Cost-based modeling for fraud and intrusion detection: Results from the JAM project’. In: *Proceedings of the DARPA Information Survivability Conference and Exposition*. Volume 2. IEEE. 2000, pages 130–144.

[107] N. Sultana, N. Chilamkurti, W. Peng and R. Alhadad. ‘Survey on SDN based network intrusion detection system using machine learning approaches’. In: *Peer-to-Peer Networking and Applications* 12.2 (2019), pages 493–501.

[108] G. G. Sundarkumar and V. Ravi. 'Malware Detection by Text and Data Mining'. In: *Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research*. IEEE. 2013, pages 1–6.

[110] S. A. Tabatabaei, X. Lu, M. Hoogendoorn and H. A. Reijers. 'Identifying Patient Groups based on Frequent Patterns of Patient Samples'. In: *Proceedings of the IEEE International Conference on E-health Networking, Application & Services*. IEEE. 2019, pages 1–6.

[111] B. A. Tama and K.-H. Rhee. 'An In-depth Experimental Study of Anomaly Detection Using Gradient Boosted Machine'. In: *Neural Computing and Applications* 31.4 (2019), pages 955–965.

[112] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani. 'A Detailed Analysis of the KDD CUP 99 Data Set'. In: *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications*. IEEE. 2009, pages 1–6.

[113] D. M. Tax and R. P. Duin. 'Uniform Object Generation for Optimizing One-class Classifiers'. In: *Journal of Machine Learning Research* 2 (2001), pages 155–173.

[115] K. Tomanek and U. Hahn. 'Semi-Supervised Active Learning for Sequence Labeling'. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. 2009, pages 1039–1047.

[116] T. P. Tran, P. Tsai, T. Jan and X. Kong. 'Network Intrusion Detection Using Machine Learning and Voting Techniques'. In: *Machine Learning* (2010), pages 267–290.

[117] A. Tsanas, M. A. Little, C. Fox and L. O. Ramig. 'Objective Automatic Assessment of Rehabilitative Speech Treatment in Parkinson's Disease'. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 22.1 (2013), pages 181–190.

[119] S. I. Wang and C. D. Manning. 'Baselines and Bigrams: Simple, Good Sentiment and Topic Classification'. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2012, pages 90–94.

[120] F. Wilcoxon. 'Some Rapid Approximate Statistical Procedures'. In: *Annals of the New York Academy of Sciences* 52.6 (1950), pages 808–814.

[122] R. Wirth and J. Hipp. 'CRISP-DM: Towards a Standard Process Model for Data Mining'. In: *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*. Volume 1. Springer-Verlag London, UK. 2000.

[124] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou and C. Wang. 'Machine Learning and Deep Learning Methods for Cybersecurity'. In: *IEEE Access* 6 (2018), pages 35365–35381.

[125] K. Yang, J. Ren, Y. Zhu and W. Zhang. 'Active Learning for Wireless IoT Intrusion Detection'. In: *IEEE Wireless Communications* 25.6 (2018), pages 19–25.

[126] D. Yarowsky. 'Unsupervised Word Sense Disambiguation Rivaling Supervised Methods'. In: *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*. 1995, pages 189–196.

[127] O. Yavanoglu and M. Aydos. 'A Review on Cyber Security Datasets for Machine Learning Algorithms'. In: *Proceedings of the IEEE International Conference on Big Data (Big Data)*. IEEE. 2017, pages 2186–2193.

[128] L. Yin, H. Wang and W. Fan. 'Active Learning Based Support Vector Data Description Method for Robust Novelty Detection'. In: *Knowledge-Based Systems* 153 (2018), pages 40–52.

[129] W. J. Youden. 'Index for Rating Diagnostic Tests'. In: *Cancer* 3.1 (1950), pages 32–35.

[130] G. U. Yule. 'On the Methods of Measuring Association Between Two Attributes'. In: *Journal of the Royal Statistical Society* 75.6 (1912), pages 579–652.

[131] M. Zamani and M. Movahedi. 'Machine Learning Techniques for Intrusion Detection'. In: *arXiv:1312.2177* (2013).

[132] S. Zhai, Y. Cheng, W. Lu and Z. Zhang. 'Deep Structured Energy Based models for Anomaly Detection'. In: *Proceedings of the 33rd International Conference on Machine Learning*. PMLR. 2016, pages 1100–1109.

[133] Y. Zhang, J. Wen, X. Wang and Z. Jiang. 'Semi-supervised Learning Combining Co-training with Active Learning'. In: *Expert Systems with Applications* 41.5 (2014), pages 2372–2378.

[134] Y. Zhao, X. Kong and S. Y. Philip. 'Positive and unlabeled learning for graph classification'. In: *Proceedings of the IEEE 11th International Conference on Data Mining*. IEEE. 2011, pages 962–971.

[135] A. Zheng and A. Casari. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O'Reilly Media, Inc., 2018.

[136] D. Zhou, Z. Yan, Y. Fu and Z. Yao. 'A Survey on Network Data Collection'. In: *Journal of Network and Computer Applications* 116 (2018), pages 9–23.

[137] L. Zhou and K. K. Lai. 'Benchmarking Binary Classification Models on Data Sets with Different Degrees of Imbalance'. In: *Frontiers of Computer Science in China* 3.2 (2009), pages 205–216.

[138] X. Zhu and A. B. Goldberg. 'Introduction to Semi-supervised Learning'. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 3.1 (2009), pages 1–130.

[139] X. Zhu, J. Lafferty and Z. Ghahramani. ‘Combining Active Learning and Semi-supervised Learning Using Gaussian Fields and Harmonic Functions’. In: *Proceedings of the ICML Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining.* Volume 3. 2003.

Online References

[1] ABC/wires. *Ransomware Cyberattack Hits Australia as EU Warns Victims Worldwide May Grow*. Retrieved on 15-05-2017. 2017. URL: http://www.abc.net.au/news/2017-05-14/ransomware-cyberattack-threat-lingers-as-people-return-to-work/8525554/.

[12] Centre for Aviation. *Fraud costs airlines USD1.4 billion a year. Regional airlines the fraudsters' "carriers of choice"*. Retrieved on 02-04-2019. 2011. URL: https://centreforaviation.com/analysis/reports/fraud-costs-airlines-usd14-billion-a-year-regional-airlines-the-fraudsters-carriers-of-choice-48150.

[19] Consultancy.eu. *Cost of cybercrime per incident jumps six-fold to €50,000*. ACS. 2020. URL: https://www.consultancy.eu/news/4409/cost-of-cybercrime-per-incident-jumps-six-fold-to-50000.

[24] D. Dua and C. Graff. *UCI Machine Learning Repository*. 2021. URL: http://archive.ics.uci.edu/ml.

[32] S. Glander. *Autoencoders and Anomaly Detection with Machine Learning in Fraud Analytics*. Retrieved on 01-05-2017. 2017. URL: https://shiring.github.io/machine_learning/2017/05/01/fraud.

[75] S. Morgan. *Cybercrime To Cost The World 10.5 Trillion Annually By 2025*. Cybercrime Magazine. 2020. URL: https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/.

[81] B. Myers. *The Sneaky Way Credit Card Scammers Go Unnoticed*. Retrieved on 06-02-2022. 2021. URL: https://www.fool.com/the-ascent/credit-cards/articles/the-sneaky-way-credit-card-scammers-go-unnoticed/.

[83] NATO Cooperative Cyber Defence Centre of Excellence. *Locked Shields 2017*. 2017. URL: https://ccdcoe.org/locked-shields-2017.html.

[86] V. Paxson. *The Bro Network Security Monitor*. 2017. URL: https://www.bro.org/.

[94] Rezgo Booking Software. *What is an OTA?* Retrieved on 02-04-2019. 2019. URL: https://www.rezgo.com/glossary/ota.

[95] RuleQuest. *Is See5/C5.0 Better Than C4.5?* Retrieved on 01-02-2017. 2017. URL: http://www.rulequest.com/see5-comparison.html.

[106] S. Stolfo et al. *KDD Cup 1999 Dataset*. UCI KDD repository. 1999. URL: http://kdd.ics.uci.edu.

[114] The Bro Platform. *base/protocols/conn/main.bro*. 2017. URL: https://www.bro.org/sphinx/scripts/base/protocols/conn/main.bro.html.

[121] Wired. *The Biggest Cybersecurity Disasters of 2017 So Far*. Retrieved on 01-07-2017. 2017. URL: https://www.wired.com/story/2017-biggest-hacks-so-far/.

[123] O. Wyman. *The Data Science Revolution That's Transforming Aviation*. Retrieved on 04-02-2022. 2016. URL: https://www.forbes.com/sites/oliverwyman/2017/06/16/the-data-science-revolution-transforming-aviation.

Summary

Technological advances make us increasingly connected online. Think of the growing number of social media services that fulfil the communication need in different ways. The increase in digital services is accompanied by an explosive growth in the number of network connections that are made between computers every second. When the Centrum Wiskunde & Informatica established the first European internet link in 1988, there were only a handful of connections worldwide. Today, every click on a website involves multiple links for at least four billion internet users. This means that more and more data is being generated, which is also easier to store due to digital developments.

The enormous amounts of data make it progressively difficult for a human to assign meaning to individual data points. If several users are active on different devices in a network, all kinds of connections are registered: browsing websites, scrolling through social media, video calling with friends, and so on. It is clear to the individual users what they are personally doing, but the 'human' meaning is usually not stored by the computer: the network links are not given a clear *label*. Knowledge and experience make it possible for cyber experts to find out relatively quickly to which action a connection belongs. However, there is a problem if the types of links are less common or encrypted, if the scale is increased, and if network traffic is monitored for a longer time. Simply due to time restrictions, it is not possible any more to give an interpretation to all the traffic.

Fortunately, cyber experts are not alone in this struggle. Over the past decades, *data science* has grown at lightning speed. This field is concerned with scientific methods and algorithms that extract information from all kinds of data. Often *Machine Learning* (ML) is used for this purpose. ML includes computer algorithms that use data to automatically improve themselves. Such a method applies mathematical techniques to discover patterns in *training data* and uses the acquired knowledge to discover the same patterns in previously unseen *evaluation data*. Consider the example about computer communication.

We can show an ML algorithm network data containing different types of labels. For example, the label indicates that the connection belongs to a visit to a secure website. The algorithm learns to recognise the characteristics of the benign links as well as the malicious ones, such that it can detect a cyber attack in new data without requiring human intervention. Since a computer can perform calculations extremely much faster, huge datasets can be interpreted in considerably less time.

Unfortunately, in reality it is not so easy to automatically recognise all possible cyber attacks by learning from a limited amount of network data. This is proven by the current state of cyber crime. Personal information is being stolen on a large scale and even national systems are being disrupted. In 2017, Ukraine was attacked by *NotPetya malware*, forcing banks and ministries to make large ransom payments to regain access to their data. Typically, ML has difficulty recognising new cyber attacks that it has not been able to learn from before. It does perform well at recognising data that deviates from what the model expects. People are good at assigning meaning to deviating data points by using domain knowledge. Therefore, we investigate how we can combine human knowledge and computer power to arrive at better and understandable predictions.

This thesis focuses on *molding the symbiosis between human and machine*. We consider the following three themes: *Anomaly Detection*, *Model Evaluation* and *Active Learning*. Within the topics, we propose how humans and computers can work together effectively and we demonstrate how our methods lead to better results.

Anomaly Detection

An anomaly is a data point that differs in a certain way from the rest of the data. An example is a credit card transaction of an amount that is many times higher than usual for the user. Such a data point can indicate fraud and is therefore interesting to investigate. Algorithms that can specifically detect anomalies are also widely used within *network intrusion detection*, the field that deals with finding cyber attacks in network data. In Chapters 2 and 3, we apply algorithms to detect malicious points in, respectively, real cyber data and data with online aeroplane bookings that contain fraudulent reservations.

First, we analyse how raw data can be converted into useful datasets. Data is usually not supplied in a nice, universal form. Therefore, we remove redundant or uninformative data features and create new variables by using domain knowledge and by combining or standardising other features. We also show that these new variables are important for detecting anomalies.

In addition, we demonstrate how we deal with unlabelled and partially labelled data. The aeroplane bookings are not labelled as legitimate or fraudulent. Hence, we use algorithms that recognise patterns in the data without needing the associated labels. For the network data, we only know of a few connections that they are involved in a cyber attack, but for the rest of the data points the status is unknown. This dataset is therefore partially labelled. By making certain assumptions we can use techniques that may or may not require labels.

Finally, we involve human experts in the process by having them evaluate the results. They answer the question whether the detected anomalies are indeed malicious. For the sake of completeness, we also show them non-anomalous data and ask them which data points are benign. This evaluation by domain experts is an important component in our research to explore the symbiosis between human and machine. By having the experts assess data points in a targeted manner, we obtain an indication of the predictive power of ML algorithms in an efficient way. Moreover, malicious data can be found faster.

Model Evaluation

It is difficult to determine the predictive power of an ML model when fully labelled evaluation data is insufficiently available. After all, the real labels are needed to determine to what extent the predicted labels are correct. Therefore, in Chapter 4 we develop an *evaluation metric* for *binary classification problems*. These are problems in which a data point belongs to only one of two classes: negative (‘0’) or positive (‘1’). Our new evaluation metric provides a good and robust estimate of the predictive power when no negative labels are available, so only positively labelled and unlabelled data is needed. This means that the quality of an ML model can be determined without using fully labelled data.

In Chapter 5, we consider Model Evaluation in a more fundamental way using a common problem that arises when evaluating any algorithm. To determine the predictive power, often an evaluation metric is used that gives a bounded score between, for example, 0 and 1. The higher the score, the better. Now, a value of 0.8 sounds good, but is that really the case? It is possible that the evaluation data has a certain structure that makes it easy to reach that score without learning a lot. Therefore, we introduce the *Dutch Draw* as a method to generate general, simple, but informative *baselines*. Such a baseline should be added to any form of binary Model Evaluation to put any score in perspective, and thus, make it more understandable.

Active Learning

In the Active Learning (AL) paradigm, the human expert becomes an integral part of the methodology. The data points expected to improve predictive power are labelled by the expert. This provides meaning to unlabelled data in an efficient way. AL is important in fields where labelled data is relatively scarce, such as in cyber security. First, an ML model is trained on the small amount of labelled data that is available. The model and statistical techniques are then used to select interesting unlabelled data points to present to the expert. They give the corresponding labels to the selected points which are then added to the labelled data. After that, the process repeats. AL is a concrete example of how the symbiosis between human and machine can be molded, because the algorithm repeatedly adapts itself in response to new knowledge that the domain expert gives to the model.

In Chapters 6 and 7, we develop new AL methods for network intrusion detection. The first method, *Jasmine*, has a *dynamic* selection procedure built in. Jasmine is able to adapt itself at every step such that the most interesting data is always selected for the expert. We show in Chapter 6 that Jasmine chooses data in a flexible way, unlike *static* selection procedures that select data points in a fixed way. This means that unlabelled data is given meaning in a more effective manner, allowing malicious network traffic to be discovered at an earlier stage.

In Chapter 7, we improve the Jasmine methodology and extend it to *Plusmine*. We refine the dynamic selection procedure based on the insights we gained in Chapter 6. Most importantly, we include *Automatic Classification* (AC) to the method that increases the speed of labelling. In AL, the human expert is usually the only source of labels. With AC we also provide an ML algorithm the opportunity to give its predicted labels to data points. For example, if the model is very certain about its prediction, then it can be assigned automatically without the intervention of the human expert. An important advantage of our method is that it works relatively simply and therefore requires little extra computational resources. We show that AC improves the detection of cyber attacks and provides new insights. This results in a fruitful collaboration between human and machine, such that working and computing time are optimally used.

Samenvatting

Door technologische vooruitgangen zijn wij steeds meer online met elkaar verbonden. Denk alleen maar aan de groeiende hoeveelheid sociale mediadiensten die op verschillende manieren invulling geven aan de communicatiebehoefte. De toename in digitale diensten gaat gepaard met een explosieve stijging van het aantal netwerkconnecties dat elke seconde tussen computers wordt gelegd. Toen het Centrum Wiskunde & Informatica in 1988 de eerste Europese internetverbinding tot stand bracht, was er wereldwijd nog maar sprake van een handvol connecties. Vandaag de dag gaat elke klik op een website gepaard met meerdere verbindingen en dat voor minstens vier miljard internetgebruikers. Dit betekent dat er steeds meer data gegenereerd wordt die bovendien door digitale ontwikkelingen makkelijker op te slaan is.

Het wordt door de enorme hoeveelheden data steeds lastiger voor een mens om betekenis aan individuele datapunten toe te kennen. Als meerdere gebruikers op verschillende apparaten in een netwerk bezig zijn, dan worden er allerlei connecties geregistreerd: het bezoeken van websites, scrollen door sociale media, videobellen met vrienden, enzovoort. Voor de individuele gebruiker is het duidelijk wat die persoonlijk uitvoert, maar de 'menselijke' betekenis wordt door de computer doorgaans niet opgeslagen: de netwerkverbindingen krijgen geen duidelijk *label*. Voor cyberexperts is het door kennis en ervaring mogelijk om relatief snel uit te zoeken bij welke handeling een connectie hoort. Echter is er een probleem als de soorten verbindingen minder gebruikelijk of versleuteld zijn, als de schaal vergroot wordt en als netwerkverkeer langer gemonitord wordt. Dan is het simpelweg door tijdsrestricties niet meer mogelijk om duiding te geven aan al het verkeer.

Gelukkig staan cyberexperts niet alleen in deze strijd. De laatste tientallen jaren is *data science* razendsnel gegroeid. Dit vakgebied houdt zich bezig met wetenschappelijke methodes en algoritmes die informatie uit allerlei soorten data weten te halen. Vaak wordt *Machine Learning* (ML) toegepast voor dit doel. Onder ML vallen computeralgoritmes die data gebruiken om zichzelf automatisch te verbeteren. Zo'n methode past wiskundige technieken toe om patronen in *trainingsdata* te ontdekken en gebruikt de opgedane kennis om dezelfde patronen in niet eerder geziene *evaluatiedata* te ontdekken. Denk terug aan het voorbeeld over computercommunicatie. Wij kunnen een ML-algoritme netwerkdata laten zien waar verschillende soorten gelabelde data in voorkomen. Het label geeft bijvoorbeeld aan dat de connectie hoort bij het bezoek aan een veilige website. Het algoritme leert de kenmerken te herkennen die tot de goedaardige verbindingen behoren en die tot de kwaadaardige behoren, zodat het een cyberaanval in nieuwe data kan detecteren zonder dat menselijke tussenkomst is vereist. Omdat een computer extreem veel sneller berekeningen kan uitvoeren, kunnen enorme datasets geduid worden in aanzienlijk minder tijd.

Helaas is het in de werkelijkheid niet zo eenvoudig om alle mogelijke cyberaanvallen automatisch te herkennen door te leren van een beperkte hoeveelheid netwerkdata. Dit wordt bewezen door de huidige staat van cybercriminaliteit. Op grote schaal wordt er persoonlijke informatie gestolen en worden er zelfs nationale systemen ontregeld. In 2017 werd Oekraïne aangevallen door *NotPetya-malware*, waardoor bijvoorbeeld banken en ministeries grote bedragen aan losgeld moesten betalen om weer toegang te krijgen tot hun gegevens. Doorgaans heeft ML moeite met het herkennen van nieuwe cyberaanvallen waar het nog niet eerder van heeft kunnen leren. Het is wel goed in het herkennen van data die afwijkt van wat het model verwacht. Een mens is juist goed in het toekennen van betekenis aan afwijkende datapunten door gebruik te maken van domeinkennis. Wij onderzoeken daarom op wat voor manier we menselijke kennis en computerkracht met elkaar kunnen combineren om tot betere en begrijpbare voorspellingen te komen.

Dit proefschrift staat in het teken van *het vormen van de symbiose tussen mens en machine*. Wij beschouwen de volgende drie thema's: *Anomaliedetectie*, *Modelevaluatie* en *Actief Leren*. Binnen de thema's doen wij voorstellen hoe mens en computer effectief samen kunnen werken en tonen wij aan hoe onze methodes tot betere resultaten leiden.

Anomaliedetectie

Een anomalie is een datapunt dat op een bepaalde manier afwijkt van de rest van de data. Denk aan een creditcardtransactie van een bedrag dat vele malen hoger is dan normaal voor de gebruiker. Zo'n datapunt kan duiden op fraude en is dus interessant om te onderzoeken. Algoritmes die specifiek anomalieën kunnen opsporen worden ook veel gebruikt binnen *netwerkintrusiedetectie*, het vakgebied dat zich bezighoudt met het vinden van cyberaanvallen in netwerkdata. In Hoofdstukken 2 en 3 passen wij algoritmes toe om kwaadaardige punten te kunnen detecteren in respectievelijk echte cyberdata en data met online vliegtuigboekingen waar frauduleuze reserveringen in voorkomen.

Allereerst analyseren we hoe ruwe data naar bruikbare datasets omgezet kan worden. Data wordt doorgaans niet aangeleverd in een mooie, universele vorm.

Daarom verwijderen we overbodige of niet-informatieve datakenmerken en creëren we nieuwe kenmerken door domeinkennis te gebruiken en door andere kenmerken te combineren of te standaardiseren. Tevens laten wij zien dat deze nieuwe kenmerken belangrijk zijn om anomalieën te kunnen ontdekken.

Bovendien tonen wij aan hoe wij omgaan met niet-gelabelde en partieel gelabelde data. De vliegtuigboekingen zijn niet gelabeld als legitiem of frauduleus. Daarom gebruiken wij algoritmes die patronen in de data herkennen zonder de bijbehorende labels nodig te hebben. Voor de netwerkdata weten we alleen van een aantal connecties dat ze betrokken zijn bij een cyberaanval, maar voor de rest van de datapunten is de status onbekend. Deze dataset is dus partieel gelabeld. Door bepaalde aannames te doen kunnen we technieken gebruiken die wel óf geen labels nodig hebben.

Uiteindelijk betrekken wij menselijke experts bij het proces door hen de resultaten te laten evalueren. Zij beantwoorden de vraag of de gedetecteerde anomalieën inderdaad kwaadaardig zijn. Voor de volledigheid tonen wij ze ook niet-afwijkende data met de vraag of die goedaardig is. Deze evaluatie door domeinexperts is een belangrijke component in ons onderzoek om de symbiose tussen mens en machine vorm te geven. Door de experts namelijk gericht datapunten te laten beoordelen, verkrijgen we op een efficiënte manier een indicatie van de voorspellingskracht van ML-algoritmes. Tevens kan kwaadaardige data sneller gevonden worden.

Modelevaluatie

Het is lastig om de voorspellingskracht van een ML-model te bepalen als volledig gelabelde evaluatiedata onvoldoende beschikbaar is. De echte labels zijn immers nodig om vast te kunnen stellen in hoeverre de voorspelde labels kloppen. Daarom ontwikkelen wij in Hoofdstuk 4 een *evaluatiemetriek* voor *binaire classificatieproblemen*. Dit zijn vraagstukken waarbij een datapunt tot uitsluitend één van twee klassen behoort: negatief (‘0’) of positief (‘1’). Onze nieuwe evaluatiemetriek geeft een goede en robuuste schatting van de voorspellingskracht als er geen negatieve labels beschikbaar zijn: er is dus alleen positief-gelabelde en ongelabelde data nodig. Dit betekent dat de kwaliteit van een ML-model bepaald kan worden zonder gebruik te maken van volledig gelabelde data.

In Hoofdstuk 5 beschouwen we Modelevaluatie op een meer fundamentele manier aan de hand van een algemeen probleem dat optreedt tijdens het beoordelen van een willekeurig algoritme. Om de voorspellingskracht te bepalen wordt vaak een evaluatiemetriek gebruikt die een begrensde score geeft tussen bijvoorbeeld 0 en 1. Hoe hoger de score, hoe beter. Zo klinkt een waarde van 0.8 goed, maar is dat wel zo? Het is mogelijk dat de evaluatiedata een bepaalde structuur heeft zodat die score makkelijk te behalen is zonder veel te leren. Daarom introduceren wij de *Nederlandse Trekking* (*Dutch Draw*) als methode om algemene, simpele, maar informatieve *referentiepunten* te genereren. Zo’n referentiepunt zou aan elke vorm van binaire Modelevaluatie toegevoegd moeten worden om iedere score in perspectief te plaatsen en daarmee beter begrijpbaar te maken.

Actief Leren

In het paradigma van Actief Leren (AL) wordt de menselijke expert een vast onderdeel van de methodologie. De datapunten waarvan verwacht wordt dat ze de voorspellingskracht verbeteren, worden gelabeld door de expert. Hierdoor wordt ongelabelde data op een efficiënte manier van betekenis voorzien. AL is belangrijk in vakgebieden waar gelabelde data relatief schaars is, zoals in cybersecurity. Het idee is dat allereerst een ML-model getraind wordt op de kleine hoeveelheid gelabelde data die beschikbaar is. Vervolgens worden het model en statistische technieken gebruikt om ongelabelde datapunten te kiezen die interessant zijn om voor te leggen aan de expert. Die geeft de bijbehorende labels aan de geselecteerde punten die daarna worden toegevoegd aan de gelabelde data. Hierna herhaalt het proces zich. AL is een concreet voorbeeld van hoe de symbiose tussen mens en machine vormgegeven kan worden, omdat het algoritme zichzelf herhaaldelijk aanpast naar aanleiding van de nieuwe kennis die de domeinexpert aan het model geeft.

In Hoofdstukken 6 en 7 ontwikkelen wij nieuwe AL-methodes voor netwerkintrusiedetectie. De eerste methode, *Jasmine*, heeft een *dynamische* selectieprocedure ingebouwd. Jasmine is in staat om zichzelf bij elke stap aan te passen zodat steeds de meest interessante data geselecteerd wordt voor de expert. Wij laten in Hoofdstuk 6 zien dat Jasmine op een flexibele manier data uitkiest, anders dan *statische* selectieprocedures die op een onveranderlijke manier datapunten selecteren. Dit betekent dat ongelabelde data op een meer doeltreffende manier van duiding wordt voorzien. Hierdoor kan kwaadaardig netwerkverkeer in een eerder stadium ontdekt worden.

In Hoofdstuk 7 verbeteren we de Jasmine-methodologie en breiden deze uit tot *Plusmine*. We schaven de dynamische selectieprocedure bij aan de hand van de inzichten die we in Hoofdstuk 6 hebben opgedaan. Maar het voornaamste is dat we *Automatische Classificatie* (AC) aan de methode toevoegen die de snelheid verhoogt waarmee gelabeld wordt. In AL is de menselijke expert normaliter de enige bron van labels. Met AC geven wij ook een ML-algoritme de mogelijkheid om zijn voorspelde labels aan datapunten te geven. Als het model bijvoorbeeld zeer zeker is over zijn voorspelling, dan kan die automatisch toegekend worden zonder tussenkomst van de menselijke expert. Een belangrijk voordeel van onze methode is dat het relatief simpel werkt en daardoor weinig extra rekenkracht vereist. Wij tonen aan dat AC de detectie van cyberaanvallen verbetert en nieuwe inzichten biedt. Dit levert een vruchtbare samenwerking tussen mens en machine op, zodat werk- en rekentijd optimaal benut worden.

About the Author

Jan Gerard Klein was born in Den Helder, the Netherlands, on the 15th of April 1992. He grew up in Oosterland, a small village in North Holland. In 2010, he finished his vwo education at rsg Wiringherlant in Wieringerwerf after which he studied BSc Mathematics at Vrije Universiteit (VU) Amsterdam. In the final year, he followed Psychology as a minor. Jan graduated in 2014 after completing his bachelor's thesis with the title 'Graphical models and the performance of the stable PC algorithm'. He continued to extend his mathematical knowledge by doing the Stochastics track of the MSc Mathematics. He also followed courses at Universiteit van Amsterdam and Universiteit Utrecht. To complete his Master's degree, Jan undertook an internship at Trafficlink in Uithoorn and Centrum Wiskunde & Informatica (CWI) in Amsterdam under supervision of Rob van der Mei, Sandjai Bhulai, and Daphne van Leeuwen. In his master's thesis 'Modelling incident management', Jan predicted the consequences of an accident on traffic such that cars could proactively be rerouted, and hence, the length of a future traffic jam would be minimised.

In 2016, Jan started his PhD research at CWI and the Dutch Ministry of the Interior and Kingdom Relations (MinBZK). This dissertation is the result of that research which will be defended on September 7, 2022. During his PhD trajectory, Jan assisted in teaching for the courses Experimental Design and Data Analysis and Statistical Models. Also, he supervised multiple master students in a broad variety of research topics and companies, such as Takeaway, Aon, and the governmental institution MinBZK. Moreover, Jan presented his work on several international conferences.

Jan maintains a broad interest in a vast number of subjects. Currently, he is working as a postdoctoral researcher at CWI and MinBZK to expand upon the research he performed during his PhD trajectory and to implement the methods he developed.