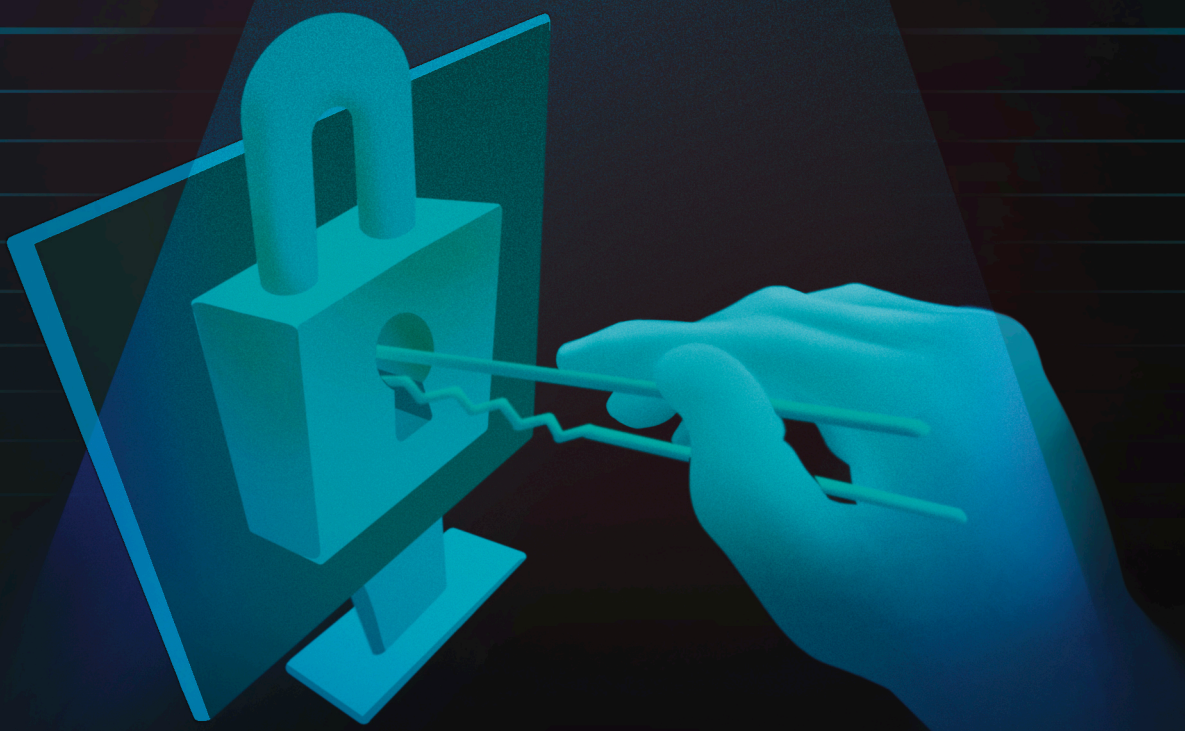
Etienne Pieter van de Bijl

From Baselines to Breakthroughs

Fundamentals and Applications of Machine Learning in Cybersecurity



Centrum Wiskunde & Informatica
Vrije Universiteit Amsterdam
Ministerie van Binnenlandse Zaken & Koninkrijksrelaties

From Baselines to Breakthroughs

Fundamentals and Applications of Machine Learning in Cybersecurity

Etienne Pieter van de Bijl

ISBN: 978-94-6473-858-2
DOI: https://doi.org/10.5463/thesis.1276

Typeset by LaTeX.
Printed by: Ipskamp Printing
Cover design by: Salim Salmi

VRIJE UNIVERSITEIT

From Baselines to Breakthroughs

Fundamentals and Applications of Machine Learning in Cybersecurity

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,
volgens besluit van de decaan
van de Faculteit der Bètawetenschappen
in het openbaar te verdedigen
op donderdag 2 oktober 2025 om 11.45 uur
in de universiteit

door

Etienne Pieter van de Bijl

geboren te Amsterdam

promotoren:	prof.dr. R.D. van der Mei
	prof.dr. S. Bhulai
promotiecommissie:	prof.dr. G.M. Koole
	prof.dr.ir. R.H.A. Lindelauf
	prof.dr. F.M. Spieksma
	prof.dr.ir. R.E. Kooij
	dr.ir. M.C. ten Thij

Voor mijn ouders Ed en Cora en zusje Francine

Preface

You see, in this world, there's two kinds of people my friend,
those with loaded guns, and those who dig. You dig.
Blondie (Clint Eastwood)
The Good, the Bad, and the Ugly, 1966

This dissertation explores how machine learning can help strengthen cybersecurity, particularly in detecting increasingly complex and evolving cyber threats. The motivation behind this research is both practical and personal: as cyberattacks become more sophisticated, so must the tools we use to defend against them. Throughout this research, a balance has been sought between fundamental contributions and practical applications. The work presented here is not just theoretical; it includes practical implementations and evaluations of machine learning techniques in cybersecurity scenarios. The goal has been to bridge the gap between academic research and practical applications, providing tools and insights that can be used by cybersecurity and datascience professionals.

I hope that the findings presented in this work inspire further research and practical advancements in securing digital systems against evolving threats.

Contents

List of Abbreviations

AL	Active Learning
DD	Dutch Draw
DO	Dutch Oracle
DS	Dutch Scaler
DSPI	Dutch Scaler Performance Indicator
DT	Decision Tree
FN	(Number of) False Negative(s)
FP	(Number of) False Positive(s)
GNB	Gaussian Naive Bayes
ID	Intrusion Detection
IDS	Intrusion Detection System
KNN	k-Nearest Neighbors
LR	Logistic Regression
ML	Machine Learning
NIDS	Network Intrusion Detection System
RF	Random Forest

SVM Support Vector Machine

TL Transfer Learning

TN (Number of) True Negative(s)

TP (Number of) True Positive(s)

1 General Introduction

Nowadays, we rely heavily on online services for a wide range of daily activities. From sending emails to managing bank accounts, their uninterrupted functionality is essential for individuals, businesses, and institutions. However, with this growing reliance is a parallel rise in the threat of *cyberattacks*. Malicious actors exploit system vulnerabilities through phishing, malware, and viruses, disrupting operations and compromising sensitive data [1]. As a result, the need for robust *cybersecurity* measures has never been more critical to safeguarding and maintaining trust in our digital ecosystem.

Commonly encountered data types in cybersecurity include network traffic logs, user behavior patterns, and system event logs, which provide valuable insights into the activities occurring within a digital environment [2]. These datasets are instrumental in identifying potential security breaches. However, working with this type of data presents several challenges. Newly collected data is *unlabeled*, meaning there is no clear indication of whether the activities are benign or malicious. Furthermore, real-time networks generate vast amounts of data that must be processed instantly to detect emerging threats, making it impractical for *cybersecurity experts* to analyze and categorize everything manually. Fortunately, *machine learning* (ML) techniques offer a promising solution to overcome these challenges.

ML, a subfield of *artificial intelligence*, enables computers to learn patterns from data and make predictions or decisions without explicit programming [3]. Two major approaches in ML are *supervised* and *unsupervised* learning. In *supervised learning*, the model is *trained* on labeled data, with each input paired with a known output. This category includes *regression*, which predicts continuous

values (e.g., forecasting house prices), and *classification*, which categorizes inputs into discrete labels (e.g., detecting spam emails). In contrast, *unsupervised learning* works with unlabeled data, where the model identifies hidden patterns or structures, such as in *clustering* or *dimensionality reduction* tasks.

This dissertation presents fundamental and applied research on ML in cybersecurity, positioning ML as a cornerstone of modern cyberdefense methods. Specifically, the dissertation is structured into three main sections that reflect this: *baseline methods*, *intrusion detection*, and *learning and breakthroughs*. Each section tackles distinct challenges and presents corresponding research questions to advance the current cybersecurity state-of-the-art. Section 1.1 lays the foundation with baseline methods for classification tasks, while Section 1.2 delves into the critical area of intrusion detection, applying ML techniques to detect cyber threats. Finally, in Section 1.3, we explore techniques designed to overcome the limitations of traditional ML methods, pushing the boundaries of ML in cybersecurity applications. The contributions in each section highlight ongoing advancements in the field.

1.1 Baseline Methods

This section examines binary classification and addresses fundamental challenges in evaluating model performance. First, we introduce the foundational concepts of binary classification. Second, we present a method that provides a baseline for binary performance metrics. Finally, we demonstrate why this baseline is the most suitable choice compared to other methods with the same characteristics.

1.1.1 Binary Classification

In a binary classification problem, the objective is to develop a model that learns to distinguish between two classes based on input data. Each instance i in our dataset comprises an input feature vector $\mathbf{x}_i$ and a corresponding label y_i that takes a value of either 0 (negative) or 1 (positive). Collectively, all instances form a dataset represented by the feature matrix $\mathbf{X}$ and the label vector $\mathbf{y}$. Specifically, M denotes the total number of instances, with $P \geq 0$ and $N \geq 0$ such that $P + N = M$.

To evaluate the model's performance, we compare the predicted labels $\hat{\mathbf{y}}$ with the actual labels $\mathbf{y}$. This comparison yields four basic counts: the number of true positives (TP), true negatives (TN), false negatives (FN), and false positives (FP). These counts are used to compute various performance metrics to quantify the model's performance. For instance, the number of predicted positives is given by

$\hat{P} = \text{TP} + \text{FP}$, while the number of predicted negatives is $\hat{N} = \text{TN} + \text{FN}$. Another example, the F_1 score combines them as follows:

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

A higher F_1 score typically means that the classifier's performance is 'better'. However, the question is, when is 'better' good enough? A *baseline* value is required for this purpose.

1.1.2 Benchmarking Performance Scores

Evaluating models requires a *baseline*: a point of reference to compare evaluation scores with, but selecting the right baseline is challenging. State-of-the-art models are often hard to reproduce due to computational demands or unavailable code. Training a classifier, such as a neural network, requires lots of data and computational power. Dummy classifiers do not always give informative values. This emphasizes the challenge of finding an informative, easy-to-compute baseline for model evaluation.

Research Question 1:
Can we define a universal baseline for binary classification evaluation metrics that serves as a clear benchmark for (newly) developed ML methods?
This question is the focus of Chapter 2.

Contributions: We introduce the *Dutch Draw* (DD), a universal baseline method for evaluating the performance of binary classification models. The DD baseline provides a standardized, theoretically derived reference point for many evaluation metrics, ensuring an insightful comparison across models. First, the DD baseline is *general*, as it applies to any binary classification problem regardless of the domain. Second, it is *simple*, because it requires no training or parameter tuning and is computed instantly. Third, it is *informative*, as it represents the optimal performance achievable without leveraging feature information, setting a clear benchmark for any learning-based method. We derive the DD baseline for commonly used evaluation measures and show that it often simplifies to (almost) always predicting a single class. By establishing this baseline, we provide a robust foundation for assessing model performance, promoting more rigorous and interpretable evaluations in ML research.

1.1.3 Optimal Input-Independent Baseline

Many state-of-the-art methods are complex and require fine-tuning, making comparisons challenging. For instance, if a model is outperformed by a simple method

that does not consider any feature information, it raises concerns about whether the model is truly learning useful patterns. It is essential to have a simple, *input-independent* baseline that can help evaluate whether a new model performs better than a trivial solution.

Besides the DD classifier, other input-independent classifiers exist that do not rely on feature information. These classifiers also provide a baseline against which models can be compared. The challenge lies in selecting the most appropriate input-independent classifier for comparison, ensuring it is simple and general while offering meaningful insights into a model's performance.

Research Question 2:
Which input-independent baseline is best suited for evaluating binary classification models, and what criteria determine its optimality for different evaluation metrics?
This question is the focus of Chapter 3.

Contributions: We examine binary baseline methods that do not rely on feature values and analyze which provides the best benchmark and why. Identifying the optimal baseline simplifies a key decision in model evaluation. We prove that the DD baseline is the best input-independent classifier for all positionally invariant evaluation metrics, assuming that samples are randomly shuffled. This establishes the DD baseline as the optimal choice under these intuitive conditions, making it a strong candidate for practical use.

1.2 Intrusion Detection

In this section, we introduce the role of an *intrusion detection system* (IDS) in cybersecurity and their techniques for detecting cyberattacks. First, we introduce IDS as a security mechanism, outlining the key characteristic differences. Next, we examine how ML can enhance IDS, particularly in detecting previously unseen attack variants. We present a method to improve the detection of novel cyber threats by optimizing dataset construction and classifier training.

1.2.1 Intrusion Detection Systems

An IDS serves as the 'burglar alarm' of computer security [4]. Scarfone and Mell [5] define *intrusion detection* (ID) as *"the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices"*. An IDS is software that automates monitoring and analysis on a computer or network.

Unlike an *intrusion prevention system* (IPS), which detects and blocks intrusions, an IDS focuses solely on detection. Thus, an IDS can be considered a component of an IPS. The two main IDS deployment systems are *host-based* (HIDS) and *network-based* (NIDS) [4].

A HIDS operates on a single machine and monitors it for suspicious activity. The detection software installed on the host is known as an *agent*. This system is often deployed on critical systems, such as publicly accessible servers or those storing sensitive data. In contrast, a NIDS monitors network traffic across specific segments or devices. A key advantage of a NIDS over a HIDS is its broader scope, as it analyzes traffic across multiple devices rather than being confined to a single host. It enables a NIDS to detect attacks that target multiple hosts or propagate through the network, offering a more comprehensive threat perspective. The two main detection approaches are *signature-based* detection and *anomaly* detection.

Signature-based detection, also known as misuse or knowledge-based detection, identifies threats by comparing observed events against a database of predefined signatures of known attacks. In contrast, anomaly-based or behavior-based detection assesses events relative to typical system behavior, flagging those significantly deviating from the norm. This approach assumes intrusions differ from regular activity, making them detectable as anomalies. Its primary advantage lies in its ability to identify previously unknown attacks.

1.2.2 Detecting Novel Cybervariants

Using ML for ID is widely studied, but its operational deployment remains limited [6]. While ML algorithms are effective at detecting patterns in large datasets [7], they struggle to identify meaningful outliers [6], despite modern cyberattacks being large and varied. Two key issues exist in anomaly-based ML research for ID [8], [9]. First, performance is often tested on outdated datasets [10], which do not reflect modern network traffic or real-time environments. Second, the ability of supervised learning methods to detect novel attack variants has not been adequately explored. Most tests are conducted in closed or open-world settings without addressing novel cyberattack variants.

Research Question 3:
To what extent can supervised binary classifiers accurately detect novel variants of application-layer cyberattacks, and how does the feature set influence their generalization to unseen threats?
This question is the focus of Chapter 4.

Contributions: We introduce a procedure to construct ML-ready network ID datasets by integrating *transport*, *network*, and *application*-layer information us-

ing *Zeek* [11]. This allows for a more comprehensive feature set compared to traditional datasets. We analyze the ability of binary classifiers to detect novel variants of *web*, *denial-of-service*, and *distributed denial-of-service* attacks. Results reveal that while classifiers can generalize to unseen variants in some cases, detection performance is not symmetric across variants. Furthermore, increasing the number of known variants in training does not automatically improve novel attack detection. Instead, selecting an appropriate combination of training attacks and classifier models leads to better generalization, highlighting the importance of constructing a good dataset.

1.3 Learning and Breakthroughs

In this section, we examine the learning perspective of ML in greater depth. First, we explore how learning can be quantified beyond standard performance metrics. Next, we introduce two techniques that enhance a model's learning capability: *active learning* (AL) and *transfer learning* (TL). Finally, we discuss a method that integrates these approaches to improve classifier performance, particularly in scenarios with limited labeled data.

1.3.1 Quantifying Learning

"How much did my model actually learn?" This fundamental question should be addressed in any ML model's development. But what does *learning* mean in this context? Mitchell [3] defines it as using experience (data) to improve performance on a task. In classification, this means learning a mapping from inputs to labels to make accurate predictions on new data. Ensuring this goal is met before deployment is crucial, but how can we measure it?

Classifier performance is typically expressed through *performance metrics* like the F_β *score* and *accuracy* [12], [13]. Performance scores need a meaningful frame of reference. For example, how do we quantify learning if a classifier achieves an F_1 score of 0.9 but its baseline is 0.89? Additionally, if data quality issues make an ideal F_1 score of 1.0 unattainable, what does that imply?

Research Question 4:
How can we quantify how much a binary classification model has actually learned beyond traditional performance metrics?
This question is the focus of Chapter 5.

Contributions: We introduce the *Dutch Scaler*, a novel performance indicator for binary classification models that quantifies learning by contextualizing empirical metric scores with performance bounds. The DS employs the DD baseline

as a lower reference point and introduces the *Dutch Oracle* to represent the expected performance of an optimal classifier. The *Dutch Scaler performance indicator* then expresses the relative contribution of these components, offering a more meaningful assessment of learning. We derive closed-form expressions for Dutch Scaler scores across multiple performance metrics, categorize their functional properties, and provide visualization techniques for interpretation. Furthermore, we contrast DS with an existing indicator, demonstrating its advantages in comparative classifier evaluation. Finally, we provide a Python repository, making DS accessible for practical use in model assessment.

1.3.2 Active Learning

AL is a human-in-the-loop ML approach where an expert, or an '*oracle*', labels selected instances [14]. Labeling large datasets can be costly (e.g. time-consuming) and may introduce redundancy or noise. AL aims to optimize this process by selecting only the most informative instances for labeling. This allows for developing high-accuracy classifiers while minimizing labeling effort [15]. AL typically follows one of two querying scenarios: *stream-based*, where instances are evaluated as they arrive, or *pool-based*, where a subset of candidate instances is presented to the expert. Different query strategies can be used to identify informative instances. Data-dependent strategies include, for example, *uncertainty* sampling [16], selecting instances on which a model is least certain, and *anomalous* sampling [17], choosing the most anomalous instances.

1.3.3 Transfer Learning

Traditional ML and data mining algorithms rely on the assumption that the distribution of labeled and unlabeled training data is the same, but this assumption is not always valid [18]. TL addresses this challenge by developing algorithms that can handle discrepancies between tasks or domains. The core idea is to extract knowledge from one or more *source tasks* and apply it to a different *target task*.

Generally speaking, there are three techniques to perform TL. First, *instance-based* TL aims at selecting relevant source/target instances to accomplish the target task. For instance, instance-based TL is used in domains like *image classification*, where selecting relevant source images can improve performance on a target dataset [19]. Second, *parameter-based* TL is where the parameters of a model are trained on one task and used on another task. This technique is commonly applied in *deep learning*, where pre-trained models on large datasets are fine-tuned for specific tasks [20]. Finally, the last approach is the *feature-based* approach, where source and target datasets are projected to a 'good' feature representation. For example, feature-based TL is used in domains like *speech recognition*, where

spoken language should be embedded to another space [21]. These three approaches each offer different ways to make TL work better, depending on what the task and data look like.

1.3.4 Overcoming the Cold-Start Phenomenon

AL reduces effort by focusing on the most informative instances [15] but could suffer from the *cold-start* phenomenon [22]; it needs an initial labeled set to make a data-dependent selection. This is especially challenging in ID, where malicious events are scarce. Publicly available ID datasets can help [23], but discrepancies between external and real-world traffic may introduce bias. TL addresses this by adapting knowledge from a *source* to a *target* domain [18], with *domain adaptation* (DA) bridging data distribution gaps.

Research Question 5:
How can we overcome AL's cold-start phenomenon in ID while improving classifier performance with minimal labeled data?
This question is the focus of Chapter 6.

Contributions: We propose ULTRA, a novel framework that integrates TL into AL to overcome the cold-start problem in ID. By embedding source and target datasets into a shared feature space using an improved feature-based TL technique called *semi-supervised transfer component analysis*, ULTRA enables classifiers to leverage external labeled data while adapting to new network environments. Additionally, our iterative weighing mechanism prioritizes informative instances. We demonstrate ULTRA's effectiveness across four ID datasets, showing that it significantly improves classifier performance with minimal labeled data.

1.4 Publications of the Author

Publications contained in this dissertation:

- 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 classification problems", *Journal of Applied Probability*, volume 62, number 2, pages 475-493, 2025. [24]
- J. Pries, E.P. van de Bijl, J.G. Klein, S. Bhulai, and R.D. van der Mei, "The optimal input-independent baseline for binary classification: The Dutch Draw", *Statistica Neerlandica*, volume 77, number 4, pages 543-554, 2023. [25]
- E.P. van de Bijl, J.G. Klein, J. Pries, R.D. van der Mei, and S. Bhulai, "Detecting novel application layer cybervariants using supervised learning", *International Journal on Advances in Security*, volume 15, number 3 & 4, pages 75-85, 2022. [26]
- E.P. van de Bijl, J.G. Klein, J. Pries, S. Bhulai, and R.D. van der Mei, "The Dutch Scaler performance indicator: How much did my model actually learn?", *Journal of Classification*, early access, pages 1-21, 2025. [27]
- E.P. van de Bijl, S. Bhulai, and R.D. van der Mei, "ULTRA: Utilizing transfer learning to overcome the active learning cold-start phenomenon in network intrusion detection", submitted for publication. [28]

Publication not contained in this dissertation:

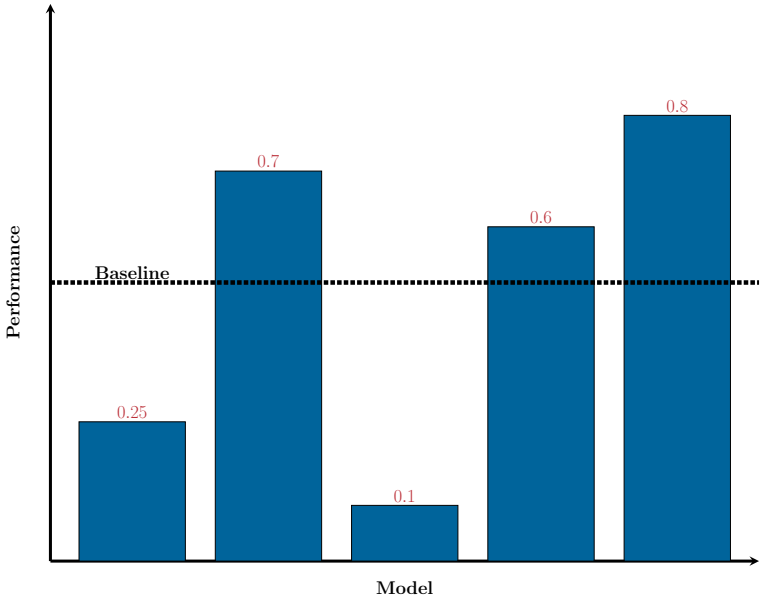
- B.E. van Leeuwen, A.W. Gansekoele, J. Pries, E.P. van de Bijl, and J.G. Klein, "Explainable kinship: A broader view on the importance of facial features in kinship recognition", *International Journal on Advances in Life Sciences*, volume 14, pages 89-99, 2022. [29]

Python implementations:

- The Dutch Draw [30].
- NIDS [31].
- The Dutch Scaler [32].
- ULTRA [33].

Part I

Baseline Methods



2

The Dutch Draw: Constructing a Universal Baseline for Binary Classification Problems

Contents

Based on [24]:
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 classification problems", *Journal of Applied Probability*, volume 62, number 2, pages 475-493, 2025.

Abstract

Novel prediction methods should always be compared to a baseline to determine their performance. Without this frame of reference, the performance score of a model is basically meaningless. What does it mean when a model achieves an F_1 of 0.8 on a test set? A proper *baseline* is, therefore, required to evaluate the 'goodness' of a performance score. Comparing results with the latest state-of-the-art model is usually insightful. However, being state-of-the-art is dynamic as newer models are continuously developed. Contrary to an advanced model, it is also possible to use a simple dummy classifier. However, the latter model could be beaten too easily, making the comparison less valuable. Furthermore, most existing baselines are stochastic and need to be computed repeatedly to get a reliable expected performance, which could be computationally expensive. In this chapter, we present a universal baseline method for all *binary classification* models, named the *Dutch Draw*. This approach weighs simple classifiers and determines the best classifier to use as a baseline. Theoretically, we derive the Dutch Draw baseline for many commonly used evaluation measures and show that in most situations, it reduces to (almost) always predicting either zero or one. Summarizing, the Dutch Draw baseline is: (1) *general*, as it is applicable to any binary classification problem; (2) *simple*, as it can be quickly determined without training or parameter-tuning; and (3) *informative*, as insightful conclusions can be drawn from the results. The Dutch Draw baseline serves two purposes. First, it is a robust and universal baseline that enables comparisons across research papers. Second, it provides a sanity check during the prediction model's development process. When a model does not outperform the Dutch Draw baseline, it is a major warning sign.

2.1 Introduction

A typical data science project can be roughly simplified into the following steps: (1) comprehending the problem context, (2) understanding the data, (3) preparing the data, (4) modeling, (5) evaluating the model, and (6) deploying the model [34]. Before deploying a new model, it should be tested to determine whether it meets certain predefined outcome criteria. A *baseline* plays an essential role in this evaluation, as it gives an indication of the actual performance of a model.

However, which method should be selected to construct a baseline? A good baseline is desirable, but what explicitly makes a baseline 'good'? Comparing results with the latest state-of-the-art model is usually insightful. However, being state-of-the-art is dynamic as newer models are continuously developed. The reproducibility of such a model is also often a problem because code is not published or large amounts of computational resources are required to retrain the model. Furthermore, most existing baselines are stochastic and need to be computed repeatedly to get a reliable expected performance, which could be computationally expensive. These aspects make comparing older results with newer research hard or even impossible. Nevertheless, it is important to stress that the comparison with a state-of-the-art model still has merit. However, we are pleading for an *additional* universal baseline that can be computed quickly (without the need for training) and can make it possible to compare results across research domains and papers. With that aim in mind, we outline three principal properties that any universal baseline construction method should have: *generality*, *simplicity*, and *informativeness*.

Generality: In research, a new model is commonly compared to a limited number of existing models used in the same field. Although these are usually carefully selected, they are still subjectively chosen. Take binary classification, in which the objective is to label each observation as either zero or one. Here, we could already select a decision tree [35], random forest [36], variants of naive Bayes [37], k-nearest neighbors [38], support vector machine [39], neural network [40], or logistic regression model [41] 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. We could not take a decision tree used for determining bankruptcy [35] and use it as a baseline for a pathological voice detection problem [42]. At least structural adaptations and retraining are necessary. A good standard baseline should be applicable to all binary classification problems, irrespective of the domain.

Simplicity: A universal baseline should not be too complex. Unfortunately, it is hard to determine whether a baseline is too complex for a measure. Essentially,

two components are critical in our view: (1) *computational time*, and (2) *explainability*. For practical applications, the baseline should be determined relatively fast. For example, training a neural network many times to generate an average baseline or optimizing the parameters of a certain model could take too much valuable time. Secondly, if a baseline is very complex, it can be harder to draw meaningful conclusions. Is this ingeniously complicated baseline expected to outperform a new model, or is it exactly what we would expect? This leads to the last property of a good standard baseline.

Informativeness: A baseline should also be informative. When a method achieves a score higher or lower than the baseline, clear conclusions need to be drawn. Is it obvious that the baseline should be beaten? Consider the athletic event *high jump*, where an athlete needs to jump over a bar at a specific height. If the bar is set too low, anyone can jump over it. If the bar is too high, no one makes it. Both situations do not give us additional information to distinguish a professional athlete from a regular amateur. The bar should be placed at a height where the professional could obviously beat it, but the amateur can not. Drawing from this analogy, a baseline should be beaten by any developed model. If not, this should be considered a major warning sign.

This research focuses on finding a general, simple, and informative baseline for *binary classification* problems. Although we focus on these types of problems, the three properties should also hold for constructing baselines in other supervised learning problems, such as multiclass classification and regression. Two methods that immediately come to mind are *dummy classifiers* and *optimal threshold classifiers*. They could be ideal candidates for our additional universal baseline.

Dummy Classifier: A dummy classifier is a *non-learning* model that makes predictions following 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 but not always informative. The information gained by performing better than a simple dummy classifier can be zero. With the plethora of dummy classifiers, the selection of one of those classifiers is also arbitrary and questionable.

Optimal Threshold Classifier: Koyejo *et al.* [43] derived 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 $\mathbb{P}(Y = 1|X = x)$, which is the probability that the binary label Y is '1' given the features $X = x$. Lipton *et al.* [44] have a similar approach, but they only focused on the F_1 score. The conditional probabilities need to be learned from training data. However, this leads to

arbitrary selections, as a model is necessary to approximate these probabilities. It is a clever approach, but unfortunately, there is no clear-cut best approximation model for different research domains. If the approximation model is not accurate, the optimal classifier is based on wrong information, which makes it hard to draw meaningful conclusions from this approach.

Both the dummy and optimal threshold classifiers have their strengths and weaknesses. In this chapter, we introduce a novel baseline approach called the *Dutch Draw* (DD). The DD eliminates these weaknesses while keeping its strengths. The DD can be seen as a dummy classifier on steroids. Instead of arbitrarily choosing a dummy classifier, we mathematically derive which classifier has the best-expected performance from a family of classifiers. Also, this expected performance can be directly determined, making it very fast to obtain the baseline. The DD baseline is: (1) applicable to any binary classification problem, (2) reproducible, (3) simple, (4) parameter-free, (5) more informative than any single dummy baseline, and (6) an explainable minimal requirement for any new model. This makes the DD an ideal candidate for a universal baseline in binary classification.

Our contributions are as follows: (1) we introduce the DD and explain why this method produces a universal baseline that is general, simple, and informative for any binary classification problem; (2) we provide the mathematical properties of the DD for many evaluation measures and summarize them in several tables; (3) we demonstrate the usefulness of the DD baseline and how it can be used in practice to identify when models should definitely be reconsidered; and (4) we made the DD available in a Python package [30].

This chapter is organized as follows. Section 2.2 introduces the required mathematical notation and the fundamentals of binary classification. Section 2.3 presents the DD framework and derives the DD baseline. In Section 2.4, we demonstrate its practical application. Finally, we summarize and conclude in Section 2.5. All mathematical derivations used throughout this chapter are detailed in Section 2.6.

2.2 Preliminaries

Before formulating the DD, we need to introduce the necessary notation and simultaneously provide elementary information on binary classification. This is required to explain how binary models are evaluated. Then, we discuss how evaluation measures are constructed for binary classification and examine the most commonly used ones.

2.2.1 Binary Classification

The goal of *binary classification* is to learn (from a dataset) the relationship between the input variables and the binary output variable. When the dataset consists of $M \in \mathbb{N}\backslash\{0\}$ observations, let $\mathcal{M} := \{1, \ldots, M\}$ be the set of observation indices. Each instance $\mathbf{x}_i$, where $i \in \mathcal{M}$ is the index, has $K \in \backslash\{0\}$ 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 $\mathbf{y} = (y_1, \ldots, y_M) \in \{0, 1\}^M$ 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 by definition, $P + N = M$ must hold.

2.2.2 Evaluation Measures

An *evaluation measure* quantifies the prediction performance of a model. We categorize the evaluation measures into two groups: *base measures* and *performance metrics* [45]. 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). Performance metrics are a function of one or more of those four base measures. To shorten notation, let $\hat{P} := \text{TP} + \text{FP}$ and $\hat{N} := \text{TN} + \text{FN}$ denote the number of positively and negatively predicted instances, respectively.

All considered base measures and performance metrics are shown in Table 2.1. Also, their abbreviations, possibly alternative names, definitions, and corresponding codomains are presented. For the F_β score, $\beta \in (0, \infty)$ is selected such that *true positive rate* (TPR) is considered β times as important as the *positive predictive value* (PPV). The 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 2.3, the case-specific codomains are taken into account when we discuss the evaluation measures in more detail. Some performance metrics, such as the *false negative rate* and *false discovery rate*, are omitted, as they can easily be derived from the results of other measures. Finally, note that the list is not exhaustive but contains most of the commonly used evaluation measures.

Ill-Defined Measures: Not every evaluation measure is well-defined. Often, the problem occurs due to division by zero. For example, the TPR is undefined whenever $P = 0$. Therefore, we have made assumptions for the allowed val-

Table 2.1: Definitions and codomains of evaluation measures/metrics

Measure	Definition	Codomain
True positives (TP)	TP	$\mathbb{N}$
True negatives (TN)	TN	$\mathbb{N}$
False negatives (FN)	FN	$\mathbb{N}$
False positives (FP)	FP	$\mathbb{N}$
True positive rate (TPR), recall, sensitivity	$\mathrm{TPR} = \frac{\mathrm{TP}}{P}$	$[0, 1]$
True negative rate (TNR), specificity, selectivity	$\mathrm{TNR} = \frac{\mathrm{TN}}{N}$	$[0, 1]$
Positive predictive value (PPV), precision	$\mathrm{PPV} = \frac{\mathrm{TP}}{\hat{P}}$	$[0, 1]$
Negative predictive value (NPV)	$\mathrm{NPV} = \frac{\mathrm{TN}}{\hat{N}}$	$[0, 1]$
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/index (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)	$\mathrm{MCC} = \frac{\mathrm{TP}\cdot\mathrm{TN}-\mathrm{FP}\cdot\mathrm{FN}}{\sqrt{\hat{P}\cdot\hat{N}\cdot P\cdot N}}$	$[-1, 1]$
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)	$\mathrm{PT} = \frac{\sqrt{\mathrm{TPR}\cdot(1\text{ - }\mathrm{TNR})}-(1\text{ - }\mathrm{TNR})}{\mathrm{TPR}-(1\text{ - }\mathrm{TNR})}$	$[0, 1]$
Threat score (TS), critical success index	$\mathrm{TS} = \frac{\mathrm{TP}}{P+\mathrm{FP}}$	$[0, 1]$

ues of P, N, $\hat{P}$, and $\hat{N}$. These are shown in Table 2.2. Some measures are not defined if P, N, $\hat{P}$ or $\hat{N}$ is equal to zero. These domain requirements are therefore necessary (always $M > 0$). One exception is the *prevalence threshold* (PT) [46], where the denominator is zero if $\mathrm{TPR} = (1 - \mathrm{TNR})$. Depending on the classifier, this situation could occur regularly. Therefore, PT is omitted throughout the rest of this research.

2.3 Dutch Draw

In this section, we introduce the DD framework and discuss how this method can provide a universal baseline for any evaluation measure. This baseline is general, simple, and informative, which is crucial for a good baseline, as we explained in Section 2.1. First, we provide the family of DD classifiers and thereafter explain how the optimal classifier generates the baseline.

Table 2.2: Assumptions on domains P, N, $\hat{P}$ and $\hat{N}$

	Domain requirement for:			
Measure	P	N	$\hat{P}$	$\hat{N}$
TP, TN, FN, FP, Acc, κ	-	-	-	-
TPR, TS	> 0	-	-	-
TNR	-	> 0	-	-
PPV	-	-	> 0	-
NPV	-	-	-	> 0
F_β, FM	> 0	-	> 0	-
J, BAcc, $G^{(2)}$	> 0	> 0	-	-
MK	-	-	> 0	> 0
MCC	> 0	> 0	> 0	> 0

2.3.1 Dutch Draw Classifiers

This research aims to provide a universal baseline for any evaluation measure in binary classification. The DD baseline comes from choosing the best DD classifier. Before discussing what 'best' entails, we have to define the DD classifier in general. This classifier generates the predictions for observations by outputting a vector of M random binary values. It is described in words as

$$\sigma_\theta(M) := \{\text{take a random set of size } \lfloor M \cdot \theta \rceil \text{ from } \mathcal{M} \text{ without replacement,} \\ \text{assign 1 to these observations and 0 to the remaining}\}.$$

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

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

with $(\mathbf{1}_E(i))_{i \in \mathcal{M}}$ the vector with ones in the positions in E and zeroes elsewhere. Note that a classifier σ_θ does not learn from the features in the data, just as a dummy classifier. The set of all DD 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 DD classifier, the number of predicted positives $\hat{P}$ depends on θ and is given by $\hat{P}_\theta := \lfloor M \cdot \theta \rceil$ and the number of predicted negatives is $\hat{N}_\theta := M - \lfloor M \cdot \theta \rceil$. Specifically, these two numbers are integers; thus, different values of θ can

lead to the same value of $\hat{Pl}_\theta$. Therefore, we introduce the parameter $\theta^* := \frac{\lfloor M \cdot \theta \rceil}{M}$ as the discretized 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 $\theta \in [0,1]$.

Next, we derive mathematical properties of the DD classifier for every evaluation measure in Table 2.1 (except PT). Note that the DD is stochastic; thus, we examine the *distribution* of the evaluation measure. Furthermore, we also determine the *range* and *expectation* of a DD classifier.

Distribution: The distributions of the base measures (see Section 2.2.2) 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 labeled positive, $\lfloor M \cdot \theta \rceil$ random observations are predicted as positive in the DD 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 labeled 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 (2.5). 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 crucial effect of the DD approach, as it reduces the formulations to only a function of a single variable. Consequently, most evaluation measures 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, $\text{TP}_\theta + \text{FN}_\theta = P$ and $\text{TN}_\theta + \text{FP}_\theta = N = M - P$. 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 $X_\theta(a,b) := a \cdot \mathrm{TP}_\theta + b$ with $a, b \in \mathbb{R}$. Additionally, let $f_{X_\theta}(a,b)$ be the probability distribution of $X_\theta(a,b)$. Then, by combining the identities, we get

$$\mathrm{TP}_\theta = \mathrm{TP}_\theta, \tag{2.1}$$

$$\mathrm{FP}_\theta = \hat{P}_\theta - \mathrm{TP}_\theta, \tag{2.2}$$

$$\mathrm{FN}_\theta = P - \mathrm{TP}_\theta, \tag{2.3}$$

$$\mathrm{TN}_\theta = N - \hat{P}_\theta + \mathrm{TP}_\theta, \tag{2.4}$$

with $\hat{P}_\theta := \lfloor M \cdot \theta \rceil$.

Example Distribution F_β Score: To show how the probability function $f_{X_\theta}(a,b)$ can directly be derived, we consider the F_β *score* [47]. It is the *weighted harmonic average* between the TPR_θ and PPV_θ. 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

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

By substituting PPV_θ and TPR_θ by their definitions (see Table 2.1) and using Equations (2.1) and (2.2), we get

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

Since PPV_θ is only defined when $\hat{P}_\theta = \lfloor M \cdot \theta \rceil > 0$ and TPR_θ is only defined 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

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

Range: The values that $X_\theta(a,b)$ can attain depend on a and b and the domain of TP_θ. Without restriction, the maximum number that TP_θ can be is P. Then, all positive observations are also predicted to be positive. 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 $\lfloor M \cdot \theta \rceil$ in this case. Hence, in general, the upper bound of 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 zero since all positive observations can be predicted as negative. 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 of the domain is $\max\{0, \lfloor M \cdot \theta \rceil - (M - P)\}$. Now, let $\mathcal{D}(\text{TP}_\theta)$ be the domain of TP_θ, then

$$\mathcal{D}(\text{TP}_\theta) := \{i \in \mathbb{N}_0 : L \leq i \leq U\}. \tag{2.5}$$

where $L = \max\{0, \lfloor M \cdot \theta \rceil - (M - P)\}$ and $U = \min\{P, \lfloor M \cdot \theta \rceil\}$. Consequently, the range of $X_\theta\,(a, b)$ is given by

$$\mathcal{R}\left(X_\theta\,(a, b)\right) := \{a \cdot i + b\}_{i \in \mathcal{D}(\text{TP}_\theta)}. \tag{2.6}$$

Expectation: The introduction of $X_\theta\,(a, b)$ allows us to write its expected value in terms of a and b. This statistic 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

$$\mathbb{E}\left[\text{TP}_\theta\right] = \frac{\lfloor M \cdot \theta \rceil}{M} \cdot P.$$

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

$$\mathbb{E}\left[X_\theta\,(a, b)\right] = a \cdot \mathbb{E}\left[\text{TP}_\theta\right] + b = a \cdot \frac{\lfloor M \cdot \theta \rceil}{M} \cdot P + b. \tag{2.7}$$

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

Example Expectation F_β Score: To demonstrate how the expectation is calculated for a performance metric, we again consider $\text{F}_\theta^{(\beta)}$. It is linear in TP_θ with $a = (1 + \beta^2)/(\beta^2 \cdot P + \lfloor M \cdot \theta \rceil)$ and $b = 0$, and so, its expectation is given by

$$\begin{aligned}\mathbb{E}\left[\text{F}_\theta^{(\beta)}\right] &= \mathbb{E}\left[X_\theta\left(\frac{1 + \beta^2}{\beta^2 \cdot P + \lfloor M \cdot \theta \rceil}, 0\right)\right] \overset{(2.7)}{=} \frac{1 + \beta^2}{\beta^2 \cdot P + \lfloor M \cdot \theta \rceil} \cdot \mathbb{E}\left[\text{TP}_\theta\right] \\ &= \frac{\lfloor M \cdot \theta \rceil \cdot P \cdot (1 + \beta^2)}{M \cdot (\beta^2 \cdot P + \lfloor M \cdot \theta \rceil)} = \frac{(1 + \beta^2) \cdot P \cdot \theta^*}{\beta^2 \cdot P + M \cdot \theta^*}.\end{aligned} \tag{2.8}$$

A full overview of the distribution and expectations of all considered base and performance metrics is given in Table 2.3. The properties are derived using a DD classifier σ_θ with $\theta^* = \frac{\lfloor M \cdot \theta \rceil}{M}$. Section 2.6 provides all the calculations to derive the corresponding distributions and expectations.

Table 2.3: Properties of performance metrics for a DD classifier

		Distribution $f_{X_\theta}(a, b)$	
Measure	**Expectation**	a	b
TP	$\theta^* \cdot P$	1	0
TN	$(1-\theta^*) \cdot (M-P)$	1	$M - P - M \cdot \theta^*$
FN	$(1-\theta^*) \cdot P$	-1	P
FP	$\theta^* \cdot (M-P)$	-1	$M \cdot \theta^*$
TPR	θ^*	$\frac{1}{P}$	0
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	$1 - \frac{P}{M}$	$\frac{1}{M \cdot (1-\theta^*)}$	$1 - \frac{P}{M \cdot (1-\theta^*)}$
F_β	$\frac{(1+\beta^2) \cdot \theta^* \cdot P}{\beta^2 \cdot P + M \cdot \theta^*}$	$\frac{1+\beta^2}{\beta^2 \cdot P + M \cdot \theta^*}$	0
J	0	$\frac{M}{P \cdot (M-P)}$	$-\frac{M \cdot \theta^*}{M-P}$
MK	0	$\frac{1}{M \cdot \theta^* (1-\theta^*)}$	$-\frac{P}{M \cdot (1-\theta^*)}$
Acc	$\frac{(1-\theta^*) \cdot (M-P) + \theta^* \cdot P}{M}$	$\frac{2}{M}$	$1 - \theta^* - \frac{P}{M}$
BAcc	$\frac{1}{2}$	$\frac{M}{2P \cdot (M-P)}$	$\frac{1}{2} - \frac{M \cdot \theta^*}{2(M-P)}$
MCC	0	$\frac{1}{\sqrt{P \cdot (M-P) \cdot \theta^* \cdot (1-\theta^*)}}$	$-\frac{\sqrt{P \cdot \theta^*}}{\sqrt{(M-P) \cdot (1-\theta^*)}}$
κ	0	$\frac{2}{P \cdot (1-\theta^*) + (M-P) \cdot \theta^*}$	$-\frac{2 \cdot \theta^* \cdot P}{P \cdot (1-\theta^*) + (M-P) \cdot \theta^*}$
FM	$\sqrt{\frac{\theta^* \cdot P}{M}}$	$\frac{1}{\sqrt{P \cdot M \cdot \theta^*}}$	0
$G^{(2)}$	-	Nonlinear in TP_θ	Nonlinear in TP_θ
TS	-	Nonlinear in TP_θ	Nonlinear in TP_θ

2.3.2 Optimal Dutch Draw Classifier

Next, we discuss how the DD baseline will ultimately be derived. An overview is presented in Figure 2.1. Starting with the definition of the DD classifiers in Section 2.3.1 and determining their expectations for commonly used measures (see Table 2.3), we are now able to identify the *optimal* DD classifier, i.e., the DD baseline. Given a performance metric and dataset, the optimal DD classifier is found by optimizing the associated expectation for $\theta \in [0, 1]$. It is natural to assume that the evaluation measures/metrics considered are maximized. Thus, metrics that are usually minimized are omitted from this chapter. The same procedure can be followed for these metrics by replacing maximizing with minimizing.

DD Baseline: The optimal DD classifiers and the corresponding DD baseline can be found in Table 2.4. For Acc_θ, note that Iverson brackets are used to simplify notation. The maximum expected score of all allowed DD classifiers, which is

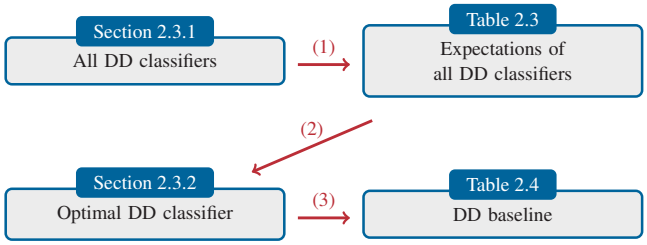


Figure 2.1: Road to the DD baseline

the DD baseline, is determined. '-' denotes in this table that no closed-form expression was found. For many performance metrics, it is optimal to predict all instances as positive or all of them as negative. In some cases, this is not allowed due to ill-defined measures. Then, it is often optimal to only predict one sample differently. For several other metrics, almost all parameter values give the optimal baseline. On the next page, we give an example to illustrate how the results of Table 2.4 are derived.

Table 2.4: Optimal DD classifier and DD baseline

Measure	$\max\{\mathbb{E}\}$	$\Theta^\star_{\max} := \arg\max\{\mathbb{E}\}$
TPR	1	$\{1\}$
TNR	1	$\{0\}$
PPV	$\frac{P}{M}$	$\Theta^* \setminus \{0\}$
NPV	$1 - \frac{P}{M}$	$\Theta^* \setminus \{1\}$
F_β	$\frac{(1+\beta^2)\cdot P}{\beta^2 \cdot P + M}$	$\{1\}$
J	0	Θ^*
MK	0	$\Theta^* \setminus \{0, 1\}$
Acc	$\max\left\{\frac{P}{M}, 1 - \frac{P}{M}\right\}$	if $P = \frac{M}{2} \rightarrow \Theta^*$ else $\rightarrow \{[P < \frac{M}{2}]\}$
BAcc	$\frac{1}{2}$	Θ^*
MCC	0	$\Theta^* \setminus \{0, 1\}$
κ	0	if $P = M \rightarrow \Theta^* \setminus \{1\}$ else $\rightarrow \Theta^*$
FM	$\sqrt{\frac{P}{M}}$	$\{1\}$
$G^{(2)}$	-	-
TS	$\frac{P}{M}$	if $P = 1 \rightarrow \Theta^* \setminus \{0\}$ else $\rightarrow \{1\}$

Example DD Baseline F_β Score: To determine the DD baseline, the extreme values of the expectation $\mathbb{E}\left[\mathrm{F}_\theta^{(\beta)}\right]$ need to be identified. To do this, examine the following function $f : [0, 1] \to [0, 1]$ defined as follows:

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

The relationship between f and $\mathbb{E}\left[\mathrm{F}_\theta^{(\beta)}\right]$ is given as $f(\lfloor M\cdot\theta\rceil/M) = \mathbb{E}\left[\mathrm{F}_\theta^{(\beta)}\right]$. To find the extreme values, we have to look at the derivative of f. This is given by

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

It is strictly positive for all t in its domain. Thus, f is strictly increasing in t. This means $\mathbb{E}\left[\mathrm{F}_\theta^{(\beta)}\right]$ 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 maximum expectation of $\mathrm{F}_\theta^{(\beta)}$ is as follows:

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

Note that $\lfloor M\cdot\theta\rceil > 0$ is a restriction for $\mathrm{F}_\theta^{(\beta)}$; hence, the maximum is taken over the interval $[1/(2M), 1]$. Furthermore, the optimization value θ_{max} is given by

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

Following this reasoning, the discrete form θ^*_{max} is given by

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

This implies that predicting everything positive yields the largest $\mathbb{E}\left[\mathrm{F}_\theta^{(\beta)}\right]$.

Selecting Performance Metrics: The expectations of the DD given in Table 2.4 and the codomains of each performance metric given in Table 2.1 indicate that DD baseline values are identical to the expected score of the 'optimal' model

for some performance metrics, like the TPR/TNR. Evaluating the performance of a classifier on one of these metrics will always give an unsatisfying result, as input-dependent classifiers can only underperform or equalize the DD baseline. In addition, evaluating a classifier on only one of the other metrics cannot give a holistic view of the performance of a classifier, as each metric weights the base measures differently. There is 'objectively' not one metric better than all other metrics. Multiple performance metrics should be checked to evaluate a classifier properly. The expectations given in Table 2.4 serve as a lookup table to validate the performance of a classifier from a holistic perspective.

Non-Linear Performance Metrics: We have shown that the DD baseline is straightforward for performance metrics that can be written in *linear* terms of TP_θ. However, there are performance metrics, such as the $\mathrm{G}_\theta^{(2)}$, where this is impossible. This could make it hard to derive a closed-form expression for the maximum expectation. Previously, we have seen in Table 2.4 that $\theta^* = 0$ or $\theta^* = 1$ was often optimal. Examining $\mathrm{G}_\theta^{(2)}$ closer shows that simply selecting $\theta^* = 0$ or $\theta^* = 1$ would result in the worst possible score. To show that the optimal parameter is less straightforward in this case, we derived the optimal $\theta^*_{\max}$ in Figure 2.2 for a fixed $M = 50$ and increasing P. This shows that $\theta = 0.5$ is not always optimal. The optimal value significantly differs when $P \ll N$ or $P \gg N$. We believe that the following reasoning can explain this: Observe that $\mathrm{G}^{(2)} = \sqrt{\mathrm{TPR} \cdot \mathrm{TNR}}$ is zero when either TPR or TNR is zero, which is the minimum score. When there are few positive labels, it must be prevented that all these samples are falsely predicted negative, which is why $\theta^*_{\max}$ is increased. The reverse holds when there are only a few negative samples. The DD baseline can still be derived for non-linear performance metrics by determining the expectations of *all* DD classifiers.

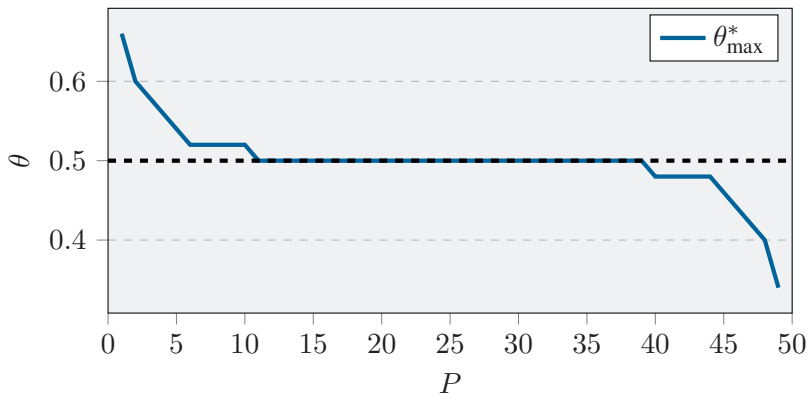


Figure 2.2: Example of non-trivial $\theta^*_{\max}$ for $\mathrm{G}_\theta^{(2)}$ with $M = 50$

2.4 Dutch Draw in Practice

Now that we have established how to derive the DD baseline, it is time to see how the DD could be used in practice.

2.4.1 Example 1: Cleveland Heart Disease

A dataset, named *Cleveland Heart Disease* dataset, is given to predict whether patients have heart disease using several feature values [48]. We randomly split the dataset into a training (90%) and a test set (10%) and choose the F_1 measure to evaluate how well a model performs. The DD baseline, derived from using M and P of the test data, immediately provides a performance reference (0.735) for any model. We train two common machine learning algorithms: a *decision tree* (DT) and the *k-nearest neighbors* (KNN) algorithm with default parameters in *scikit-learn* [49]. To estimate the expected performance, we average the results obtained from 10 different random seeds for each stochastic model. They achieve an average score of 0.727 and 0.710, respectively, which are worse than the DD baseline. This is a major warning sign. Although the DT performed better than KNN, it should still not be used. Thus, we decided to train three other models: *logistic regression* (LR), *random forest* (RF), and *Gaussian naive Bayes* (GNB), which end up performing better than the baseline. Finally, we selected the LR in practice, as this model achieves the highest score *and* beats the DD baseline. Table 2.5 shows an overview of the performance of these five models and the baseline on a set of selected performance metrics. The red scores highlight the situations where a model achieves a score inferior to the DD baseline.

Table 2.5: Comparing classifier performances to the DD baseline on the Cleveland Heart Disease dataset

		Classifier				
Metric	**DD baseline**	**DT**	**KNN**	**LR**	**RF**	**GNB**
Acc	0.581	0.710	0.710	0.903	0.871	0.839
BAcc	0.500	0.718	0.729	0.906	0.868	0.840
F_1	0.735	0.727	0.710	0.914	0.889	0.857
FM	0.762	0.730	0.719	0.915	0.889	0.857
$G^{(2)}$	0.500	0.716	0.719	0.906	0.867	0.840
J	0.000	0.436	0.457	0.812	0.735	0.679
κ	0.000	0.422	0.434	0.803	0.735	0.672
MCC	0.000	0.430	0.457	0.805	0.735	0.674
MK	0.000	0.425	0.457	0.798	0.735	0.668
NPV	0.419	0.625	0.611	0.857	0.846	0.786
PPV	0.581	0.800	0.846	0.941	0.889	0.882
TS	0.581	0.571	0.550	0.842	0.800	0.750

2.4.2 Example 2: ImageNet

The aforementioned example illustrates how the DD baseline provides insights into the performance of a set of standard machine-learning models. Nevertheless, tasks involving higher-dimensional data necessitate more advanced model architectures. Image-related classification tasks serve as a prevalent example of such complex problems, and we aim to demonstrate in this example how the DD baseline imparts valuable insights for these models tackling more demanding challenges. The *ImageNet Large Scale Visual Recognition Challenge 2012* (ILSVRC2012) offers a significant dataset comprising over 1 million images across 1,000 diverse object categories [50]. This challenge involves developing a multiclass classification model that accurately predicts which object (such as a sock, volleyball, vase, etc.) is visible in each image. We have selected a range of state-of-the-art vision models, including *DenseNet121*, *GoogLeNet*, *ResNet18*, *ResNet50*, *VGG11*, and *VGG19* to tackle this classification problem. These models are implemented and pre-trained in PyTorch [51] on the training dataset of the ImageNet dataset. Typically, the performance of these models is assessed based on top-1 or top-5 accuracy rates. However, our interest lies in identifying the categories where these models do not perform as well to test the underlying performance. The test data labels are not publicly available, but we can assess these models using the provided validation dataset, which contains 50 images per category. The outputs of the selected vision models on this dataset are a vector of probabilities representing the likelihood of the input image belonging to each of the 1,000 classes. The model assigns the class label with the highest likelihood to the instance. We calculate the four base measures for each category i by categorizing all instances of label i as 'positive' and all other labels as 'negative' in both the predicted and actual data. There are 50 images per category in the validation dataset, so the DD baseline, independently of the selected evaluation metric, is identical for each category (P = 50, M = 50,000). Table 2.6 shows the number of categories having an inferior *accuracy* score when compared to its corresponding DD baseline (0.999). As the DD baseline takes class imbalance into account, accuracy scores can be interpreted for this binary classification problem even though this metric is not robust against imbalance. It is crucial to explore categories that fail the sanity check, whether this stems from issues with the data or the models themselves. There are, in total, 24 categories where none of the selected vision models actually outperform the DD baseline, indicating underlying issues with these categories. Enhancing the performance in these weaker categories would contribute to the overall efficacy of these multiclass classification models.

Table 2.6: Number of categories dataset where vision models underperform relative to the DD baseline in the ImageNet challenge

Model	Number of underperforming categories
DenseNet121	49
GoogLeNet	97
ResNet18	94
ResNet50	43
VGG11	112
VGG19	63

2.4.3 Example 3: CelebA

In the previous example, we delved into a high-dimensional multiclass classification problem, illustrating how the DD baseline aids in the improvement of models tackling this problem. Now, we focus on another realm of high-dimensional challenges, specifically those involving multiple binary classification tasks, the so-called multitask learning problem, namely the image-related task of detecting facial attributes. The problem is to detect these attributes in images, like wearing glasses or a cap. The *CelebA* dataset can be used for this task and generative image creation [52]. The dataset consists of 100,000+ images of 10,177 identities. Each image is annotated with 40 facial attributes. To the best of our knowledge, no publicly available trained model is yet available. Fortunately, we can use pretrained models of other datasets, like the ImageNet dataset, and replace the last layer of each network with a layer, making predictions for each binary facial attribute. The approach is commonly known as transfer learning. We selected the same vision models as for the ImageNet dataset. We trained this last layer on the training dataset of CelebA based on the *weighted cross entropy loss* that corrects for imbalance in the training data and the standard *Adam optimizer* in batches of 100 images. The labels for the test dataset comprising 19,962 images are publicly available, so the trained vision models are evaluated on this data. Table 2.7 shows the results of the vision models evaluated on the test dataset of the CelebA dataset. The DenseNet121, GoogLeNet, ResNet18, ResNet50 are abbreviated as DN121, GN, RN18, and RN50, respectively. The DD baseline helps identify attributes where the transferred state-of-the-art vision models score inferior. We selected attributes where at least one of the vision models underperformed compared to the DD baseline. If the score is inferior to the DD baseline, then it is highlighted in red. In addition, we have selected the F_1 score to assess each model's performance, as the data has an attribute-dependent class imbalance. Still, similar results can be acquired when selecting another evaluation metric. Remarkably

are the attributes *Big_Lips*, *Narrow_Eyes*, *Oval_Face*, *Pointy_Nose*, and *Wearing_Necklace*, where all vision models perform inferior. Like our statement for the ImageNet dataset, data or model-related issues can cause this underperformance. Again, the DD baseline helps identify features where classification models score inferior.

Table 2.7: Performance of vision models on the CelebA dataset (F_1 score) compared to the DD baseline

	Classifier						
Attribute	**DD baseline**	**DN121**	**GN**	**RN18**	**RN50**	**VGG11**	**VGG19**
Arched_Eyebrows	0.443	0.476	0.497	0.432	0.383	0.519	0.446
Bags_Under_Eyes	0.337	0.444	0.431	0.401	0.491	0.350	0.131
Big_Lips	0.493	0.209	0.315	0.137	0.227	0.131	0.124
Brown_Hair	0.305	0.416	0.454	0.524	0.548	0.388	0.255
Bushy_Eyebrows	0.229	0.348	0.210	0.346	0.373	0.202	0.282
Double_Chin	0.087	0.325	0.317	0.267	0.220	0.127	0.071
Mouth_Slightly_Open	0.662	0.790	0.710	0.765	0.767	0.655	0.643
Mustache	0.074	0.119	0.228	0.259	0.142	0.079	0.058
Narrow_Eyes	0.259	0.201	0.081	0.122	0.112	0.018	0.016
Oval_Face	0.456	0.354	0.239	0.432	0.398	0.330	0.383
Pale_Skin	0.081	0.369	0.357	0.432	0.459	0.082	0.055
Pointy_Nose	0.444	0.257	0.218	0.387	0.330	0.208	0.282
Receding_Hairline	0.156	0.226	0.305	0.368	0.190	0.142	0.060
Rosy_Cheeks	0.134	0.430	0.333	0.350	0.438	0.103	0.117
Straight_Hair	0.347	0.384	0.321	0.399	0.540	0.383	0.234
Wearing_Earrings	0.343	0.536	0.412	0.456	0.479	0.372	0.262
Wearing_Necklace	0.242	0.051	0.036	0.037	0.109	0.038	0.018

2.5 Discussion and Conclusion

In this chapter, we introduced the DD, a new baseline methodology. The DD baseline is: (1) applicable to any binary classification problem, (2) reproducible, (3) simple, (4) parameter-free, (5) more informative than any single dummy baseline, and (6) an explainable minimal requirement for any new model. We have shown that for the most commonly used measures, the DD baseline can be theoretically determined (see Table 2.4). When the baseline cannot be derived directly, it can quickly be identified by computation. For most performance metrics, the DD baseline reduces to one of the following three cases: (1) always predicting positive or negative, (2) always predicting positive or negative, except for one instance, and (3) any DD classifier, except maybe for $\theta^* = 0$ or $\theta^* = 1$. However, there are exceptions to these three cases, as was shown with the $G_\theta^{(2)}$. This shows that the DD is not always reduced to one of the three previously mentioned cases and

does not always give straightforward results.

By introducing the DD baseline, we have simplified and improved the evaluation process of new binary classification methods. We consider it a minimum requirement for any novel model to at least beat the DD baseline. When this does not happen, the question of how much a new method has even learned from the data is raised since the DD baseline is derived from dummy classifiers. When the novel model has beaten the DD baseline, it should still be compared to a state-of-the-art method in that domain to obtain additional insights. In Section 2.4, we have shown how the DD should be used in practice. In Section 2.4.1, we show that commonly used approaches, such as *k-nearest neighbors* and a *decision tree*, can underperform. An advantage of the DD is that the expected performance can be directly derived, as opposed to other (stochastic) models, where the expected value can only be estimated through (many) experiments. Sections 2.4.2 and 2.4.3 discuss two image-related tasks where the DD baseline can identify task-related underperformance of vision models. The DD baseline can identify problematic categories in multiclass problems or specific tasks in multitask classification problems. The goal of these experiments is to show that benchmarking classification models is essential. The insights obtained from the DD baseline provide valuable perspectives that empower us to enhance and refine models. Hence, using the DD as a general, simple, and informative baseline should be the new gold standard in any binary model evaluation process. Our baseline is a stepping stone for further research, where multiple avenues should be explored. We discuss five possible research directions.

First, a wide range of other input-independent classifiers should be explored. Does the DD baseline outperform these methods, and can this be proven? If so, it would provide a solid argument to use the DD baseline over any other input-independent baseline in future applications, which is why this research question is important. This research direction motivated the development of Chapter 3.

Second, we can now determine whether a binary classification model performs better than a universal 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 quickly 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 difficult 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 the performance of these models be quantified? Is it only a slightly better model, or is it a leap forward? This research direction motivated the development of Chapter 5.

Third, our DD baseline could be used to construct new standardized evaluation measures 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 DD baseline would already be incorporated in the new measure, so comparing a score to the baseline is no longer necessary. The DD baseline can be used to scale a measure in many ways. Let $\Delta_{\max}$ and $\Delta_{\min}$ denote the maximum and minimum DD baseline, respectively. As an example, a measure μ with range $[\mu_{\min}, \mu_{\max}]$ that needs to be maximized can be rescaled by

$$\mu_{\text{rescaled}} = \begin{cases} -1 & \text{if } \mu \leq \Delta_{\min}, \\ \frac{\mu - \Delta_{\max}}{\Delta_{\max} - \Delta_{\min}} & \text{if } \Delta_{\min} \leq \mu \leq \Delta_{\max}, \\ \frac{\mu - \Delta_{\max}}{\mu_{\max} - \Delta_{\max}} & \text{else.} \end{cases}$$

Everything below the lowest DD baseline ($\Delta_{\min}$) gets value -1, because every DD classifier performs better. This should be a major warning sign. A score between $\Delta_{\min}$ and $\Delta_{\max}$ is rescaled to $[-1, 0]$. This value indicates that the performance is still worse than the best DD baseline. All scores above $\Delta_{\max}$ are scaled to $[0, 1]$. In this case, the performance of a classifier outperforms the best DD baseline.

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

Fifth, for some (less straightforward) performance metrics, the DD baseline is derived by examining the expectations of *all* DD classifiers. Thus, faster techniques should be developed for large applications. Insights could greatly improve the computation time. For example, we conjecture for $\mathrm{G}_\theta^{(2)}$ that $\theta^*_{\max} \in [0, \frac{1}{2}]$, when $P > N$ and $\theta^*_{\max} \in [\frac{1}{2}, 1]$, when $P < N$. This already reduces the search domain by half. Proving convexity could also make it easier to derive the optimal value. Decreasing the computational time could be essential for some large applications and should be investigated.

Finally, we have published the code for the DD so the reader can easily implement the baseline into their binary classification problems [30].

2.6 Appendix

This section contains the complete theoretical analysis used to gather the information presented in Section 2.3, and more specifically, Tables 2.2, 2.3, and 2.4. Each subsection is dedicated to one of the evaluation measures/metrics. The following definitions are frequently used throughout this section:

$$X_\theta(a,b) := a \cdot \mathrm{TP}_\theta + b \text{ with } a,b \in \mathbb{R},$$
$$f_{X_\theta}(a,b) := \text{probability distribution of } X_\theta(a,b).$$

An overview of the entire appendix can be viewed in Table 2.8.

Table 2.8: Overview of DD mathematical derivations

Measure	TP	TN	FN	FP	TPR	TNR	PPV	NPV	F_β	J
Section	2.6.1	2.6.2	2.6.3	2.6.4	2.6.5	2.6.6	2.6.7	2.6.8	2.6.9	2.6.10
Measure	MK	Acc	BAcc	MCC	κ	FM	$G^{(2)}$	PT	TS	
Section	2.6.11	2.6.12	2.6.13	2.6.14	2.6.15	2.6.16	2.6.17	2.6.18	2.6.19	

2.6.1 Number of True Positives

The number of *true positives* (TP_θ) is one of the four base measures introduced in Section 2.2.2. This measure indicates how many of the predicted positive observations are actually positive. Under the DD methodology, each evaluation measure can be written in terms of TP_θ. This is a measure that is naturally maximized.

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

$$\mathrm{TP}_\theta \overset{(2.1)}{=} X_\theta(1,0) \sim f_{X_\theta}(1,0).$$

The range of this base measure depends on θ. Therefore, Equation (2.6) yields the range of this measure:

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

Expectation: The expectation of TP_θ using the DD is given by

$$\mathbb{E}[\mathrm{TP}_\theta] = \mathbb{E}[X_\theta(1,0)] \overset{(2.7)}{=} \frac{\lfloor M \cdot \theta \rceil}{M} \cdot P = \theta^* \cdot P. \tag{2.9}$$

Baseline: TP_θ is a measure that is naturally maximized. The DD classifier with the highest expectation gives the DD baseline. Equation (2.9) shows that the expected value depends on the parameter θ. Therefore, the DD baseline is given by

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

The value of $\theta \in [0,1]$ maximizing the expected value is $\theta_{\max}$ and is defined as follows:

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

Equivalently, the discrete optimizer $\theta^*_{\max} \in \Theta^*$ is

$$\theta^*_{\max} \in \arg\max_{\theta^*\in\Theta^*} \mathbb{E}\left[\text{TP}_{\theta^*}\right] = \arg\max_{\theta^*\in\Theta^*} \{\theta^*\} = \{1\}.$$

2.6.2 Number of True Negatives

The number of *true negatives* (TN_θ) is also one of the four base measures and is introduced in Section 2.2.2. This base measure counts the number of negative predicted instances that are actually negative. This is also a measure that is naturally maximized.

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

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

which corresponds to Equation (2.4). Furthermore,

$$\text{TN}_\theta \overset{(2.4)}{=} X_\theta\left(1, M-P-\lfloor M\cdot\theta\rceil\right) \sim f_{X_\theta}\left(1, M-P-\lfloor M\cdot\theta\rceil\right),$$

and for its range

$$\text{TN}_\theta \overset{(2.6)}{\in} \mathcal{R}\left(X_\theta\left(1, M-P-\lfloor M\cdot\theta\rceil\right)\right).$$

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

$$\begin{aligned}\mathbb{E}\left[\text{TN}_\theta\right] &= \mathbb{E}\left[X_\theta\left(1, M-P-\lfloor M\cdot\theta\rceil\right)\right] \overset{(2.7)}{=} 1\cdot\mathbb{E}\left[\text{TP}_\theta\right] + M - P - \lfloor M\cdot\theta\rceil \\ &= \left(1-\frac{\lfloor M\cdot\theta\rceil}{M}\right)(M-P) = (1-\theta^*)(M-P).\end{aligned}$$

Baseline: TN_θ is a measure that is naturally maximized. To obtain the DD baseline, its highest expectation yields

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

The associated optimization value $\theta_{\max} \in [0,1]$ is

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

The discrete equivalent $\theta^*_{\max} \in \Theta^*$ yields

$$\theta^*_{\max} \in \arg\max_{\theta^*\in\Theta^*} \mathbb{E}\left[\text{TN}_{\theta^*}\right] = \arg\max_{\theta^*\in\Theta^*} \{1-\theta^*\} = \{0\}.$$

2.6.3 Number of False Negatives

The number of *false negative* (FN_θ) is another of the four base measures introduced in Section 2.2.2. This base measure counts the number of mistakes made by predicting instances negative while the actual labels are positive. In contrast to the previous base measure, this is a measure that is naturally minimized.

Definition and Distribution: Equation (2.3) shows that FN_θ can be expressed in terms of TP_θ:

$$\text{FN}_\theta \overset{(2.3)}{=} P - \text{TP}_\theta = X_\theta\left(-1, P\right) \sim f_{X_\theta}\left(-1, P\right),$$

and for its range

$$\text{FN}_\theta \overset{(2.6)}{\in} \mathcal{R}\left(X_\theta\left(-1, P\right)\right).$$

Expectation: As Equation (2.3) 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

$$\begin{aligned}\mathbb{E}\left[\text{FN}_\theta\right] &= \mathbb{E}\left[X_\theta\left(-1, P\right)\right] \overset{(2.7)}{=} -1 \cdot \mathbb{E}\left[\text{TP}_\theta\right] + P \\ &= \left(1 - \frac{\lfloor M\cdot\theta\rceil}{M}\right)\cdot P = (1-\theta^*)\cdot P.\end{aligned}$$

Baseline: FN_θ is a measure that is naturally minimized. The θ resulting in the lowest expectation of FN_θ determines the baselines. It is given by

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

The associated optimization value $\theta_{\min} \in [0,1]$ is

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

The discrete version $\theta^*_{\min} \in \Theta^*$ of the optimizer is given by

$$\theta^*_{\min} \in \underset{\theta^*\in\Theta^*}{\arg\min}\, \mathbb{E}\left[\mathrm{FN}_{\theta^*}\right] = \underset{\theta^*\in\Theta^*}{\arg\min} \left\{1-\theta^*\right\} = \{1\}.$$

2.6.4 Number of False Positives

The number of *false positives* (FP_θ) is the last of the four base measures. This base measure counts the number of mistakes made by predicting instances as positive while the actual labels are negative. The measure FP_θ is naturally minimized.

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

$$\mathrm{FP}_\theta \overset{(2.2)}{=} \lfloor M\cdot\theta\rceil - \mathrm{TP}_\theta = X_\theta\left(-1, \lfloor M\cdot\theta\rceil\right) \sim f_{X_\theta}\left(-1, \lfloor M\cdot\theta\rceil\right),$$

and for its range

$$\mathrm{FP}_\theta \overset{(2.6)}{\in} \mathcal{R}\left(X_\theta\left(-1, \lfloor M\cdot\theta\rceil\right)\right).$$

Expectation: As Equation (2.2) 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 follows:

$$\begin{aligned}\mathbb{E}\left[\mathrm{FP}_\theta\right] &= \mathbb{E}\left[X_\theta\left(-1, \lfloor M\cdot\theta\rceil\right)\right] \overset{(2.7)}{=} -1\cdot\mathbb{E}\left[\mathrm{TP}_\theta\right] + \lfloor M\cdot\theta\rceil \\ &= \frac{\lfloor M\cdot\theta\rceil}{M}\cdot(M-P) = \theta^*\cdot(M-P).\end{aligned}$$

Baseline: FP_θ is a measure that is naturally minimized. The DD classifier yielding the lowest expectation gives the baselines of FP_θ. Hence,

$$\min_{\theta\in[0,1]} \mathbb{E}\left[\mathrm{FP}_\theta\right] = (M-P)\min_{\theta\in[0,1]}\left(\frac{\lfloor M\cdot\theta\rceil}{M}\right) = 0.$$

The corresponding optimization value $\theta_{\min} \in [0,1]$ is

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

and its discrete version $\theta^*_{\min} \in \Theta^*$ is

$$\theta^*_{\min} \in \underset{\theta^*\in\Theta^*}{\arg\min}\, \mathbb{E}\left[\mathrm{FP}_{\theta^*}\right] = \underset{\theta^*\in\Theta^*}{\arg\min}\left\{\theta^*\right\} = \{0\}.$$

2.6.5 True Positive Rate

The *true positive rate* (TPR_θ), also called *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. It is a metric that is naturally maximized.

Definition and Distribution: The TPR_θ is commonly defined as follows:

$$\text{TPR}_\theta = \frac{\text{TP}_\theta}{P}. \tag{2.10}$$

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

$$\text{TPR}_\theta = X_\theta\left(\frac{1}{P}, 0\right) \sim f_{X_\theta}\left(\frac{1}{P}, 0\right),$$

and for its range

$$\text{TPR}_\theta \overset{(2.6)}{\in} \mathcal{R}\left(X_\theta\left(\frac{1}{P}, 0\right)\right).$$

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

$$\mathbb{E}\left[\text{TPR}_\theta\right] = \mathbb{E}\left[X_\theta\left(\frac{1}{P}, 0\right)\right] \overset{(2.7)}{=} \frac{1}{P} \cdot \mathbb{E}\left[\text{TP}_\theta\right] + 0 = \frac{\lfloor M \cdot \theta \rceil}{M} = \theta^*.$$

Baseline: TPR_θ is a measure that is naturally maximized. The DD classifier yielding the highest expectation gives the baselines of TPR_θ. Hence,

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

Furthermore, the corresponding optimization value $\theta_{\max} \in [0, 1]$ is given by

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

The discrete version $\theta^*_{\max} \in \Theta^*$ of this optimizer is

$$\theta^*_{\max} \in \arg\max_{\theta^* \in \Theta^*} \mathbb{E}\left[\text{TPR}_{\theta^*}\right] = \arg\max_{\theta^* \in \Theta^*} \{\theta^*\} = \{1\}.$$

2.6.6 True Negative Rate

The *true negative rate* (TNR_θ), also called *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. It is naturally maximized.

Definition and Distribution: The TNR_θ is commonly defined as follows:

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

Hence, $N := M - P > 0$ should hold, otherwise, the denominator is zero. By using Equation (2.4), TNR_θ can be rewritten as

$$\text{TNR}_\theta = \frac{M - P - \lfloor M \cdot \theta \rceil + \text{TP}_\theta}{M - P} = 1 - \frac{\lfloor M \cdot \theta \rceil - \text{TP}_\theta}{M - P}. \tag{2.11}$$

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

$$\text{TNR}_\theta = X_\theta \left(\frac{1}{M-P}, 1 - \frac{\lfloor M \cdot \theta \rceil}{M-P} \right) \sim f_{X_\theta} \left(\frac{1}{M-P}, 1 - \frac{\lfloor M \cdot \theta \rceil}{M-P} \right),$$

and for its range

$$\text{TNR}_\theta \overset{(2.6)}{\in} \mathcal{R} \left(X_\theta \left(\frac{1}{M-P}, 1 - \frac{\lfloor M \cdot \theta \rceil}{M-P} \right) \right).$$

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

$$\begin{aligned} \mathbb{E}[\text{TNR}_\theta] &= \mathbb{E}\left[X_\theta \left(\frac{1}{M-P}, 1 - \frac{\lfloor M \cdot \theta \rceil}{M-P} \right) \right] \\ &\overset{(2.7)}{=} \frac{1}{M-P} \cdot \mathbb{E}[\text{TP}_\theta] + 1 - \frac{\lfloor M \cdot \theta \rceil}{M-P} = 1 - \frac{\lfloor M \cdot \theta \rceil}{M} = 1 - \theta^*. \end{aligned}$$

Baseline: TNR_θ is a measure that is naturally maximized. The DD classifier yielding the highest expectation gives the baselines of TNR_θ. Hence,

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

Moreover, the optimization value $\theta_{\max} \in [0, 1]$ corresponding to this value is defined as follows:

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

The discrete version $\theta^*_{\max} \in \Theta^*$ of this optimizer is given by

$$\theta^*_{\max} \in \arg\max_{\theta^* \in \Theta^*} \mathbb{E}[\text{TNR}_{\theta^*}] = \arg\max_{\theta^* \in \Theta^*} \{1 - \theta^*\} = \{0\}.$$

2.6.7 Positive Predictive Value

The *positive predictive value* (PPV_θ), also called *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. It is commonly desirable to maximize this metric.

Definition and Distribution: The PPV_θ is commonly defined as follows:

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

By using Equations (2.1) and (2.2), this definition can be reformulated to

$$\text{PPV}_\theta = \frac{\text{TP}_\theta}{\lfloor M \cdot \theta \rceil}. \tag{2.12}$$

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

$$\text{PPV}_\theta = X_\theta \left(\frac{1}{\lfloor M \cdot \theta \rceil}, 0 \right) \sim f_{X_\theta} \left(\frac{1}{\lfloor M \cdot \theta \rceil}, 0 \right),$$

with range

$$\text{PPV}_\theta \overset{(2.6)}{\in} \mathcal{R} \left(X_\theta \left(\frac{1}{\lfloor M \cdot \theta \rceil}, 0 \right) \right).$$

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

$$\mathbb{E}\left[\text{PPV}_\theta\right] = \mathbb{E}\left[X_\theta \left(\frac{1}{\lfloor M \cdot \theta \rceil}, 0 \right) \right] \overset{(2.7)}{=} \frac{1}{\lfloor M \cdot \theta \rceil} \cdot \mathbb{E}\left[\text{TP}_\theta\right] + 0 = \frac{P}{M}.$$

Baseline: The baseline of the PPV_θ is determined by searching for the DD classifier yielding the highest expectation, as PPV_θ is a metric that is naturally maximized. It is given by

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

The expectation does not depend on θ. Hence, the optimization value $\theta_{\max}$ is in the set of all allowed values for θ

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

Consequently, the discrete version $\theta^*_{\max}$ is in the set of all allowed discrete values

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

2.6.8 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. This metric is naturally maximized.

Definition and Distribution: The NPV_θ is commonly defined as follows:

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

With the help of Equations (2.3) and (2.4), this definition can be rewritten as

$$\text{NPV}_\theta = 1 - \frac{P - \text{TP}_\theta}{M - \lfloor M \cdot \theta \rceil}. \tag{2.13}$$

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

$$\begin{aligned}\text{NPV}_\theta &= X_\theta \left(\frac{1}{M - \lfloor M \cdot \theta \rceil}, 1 - \frac{P}{M - \lfloor M \cdot \theta \rceil}\right) \\ &\sim f_{X_\theta}\left(\frac{1}{M - \lfloor M \cdot \theta \rceil}, 1 - \frac{P}{M - \lfloor M \cdot \theta \rceil}\right),\end{aligned}$$

with range

$$\text{NPV}_\theta \overset{(2.6)}{\in} \mathcal{R}\left(X_\theta \left(\frac{1}{M - \lfloor M \cdot \theta \rceil}, 1 - \frac{P}{M - \lfloor M \cdot \theta \rceil}\right)\right).$$

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

$$\begin{aligned}\mathbb{E}\left[\text{NPV}_\theta\right] &= \mathbb{E}\left[X_\theta\left(\frac{1}{M - \lfloor M \cdot \theta \rceil}, 1 - \frac{P}{M - \lfloor M \cdot \theta \rceil}\right)\right] \\ &\overset{(2.7)}{=} \frac{1}{M - \lfloor M \cdot \theta \rceil} \cdot \mathbb{E}\left[\text{TP}_\theta\right] + 1 - \frac{P}{M - \lfloor M \cdot \theta \rceil} = 1 - \frac{P}{M}.\end{aligned}$$

Baseline: The baseline of the NPV_θ is determined by searching for the DD classifier yielding the highest expectation, as NPV_θ is a metric that is naturally maximized. It is given by

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

The expectation does not depend on θ. Consequently, the optimization value θ_{max} is in the set of all allowed values for θ. Thus,

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

This also means the discrete form θ^*_{max} of the optimizer is also in the set of all allowed discrete values

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

2.6.9 F_β score

The F_β *score* ($\text{F}_\theta^{(\beta)}$) was introduced by Chinchor [47]. It is the weighted harmonic average between TPR_θ and PPV_θ. These two performance measures are discussed extensively in Sections 2.6.5 and 2.6.7. 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 is a metric that is naturally maximized.

Definition and Distribution: The F_β score is commonly defined as follows:

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

By using the definitions of TPR_θ and PPV_θ in Equations (2.10) and (2.12), $\text{F}_\theta^{(\beta)}$ can be formulated in terms of the base measures as follows:

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

Equations (2.1) and (2.2) 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

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

with range

$$\mathrm{F}_\theta^{(\beta)} \overset{(2.6)}{\in} \mathcal{R}\left(X_\theta\left(\frac{1+\beta^2}{\beta^2 \cdot P + \lfloor M \cdot \theta \rceil}, 0\right)\right).$$

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

$$\begin{aligned}\mathbb{E}\left[\mathrm{F}_\theta^{(\beta)}\right] &= \mathbb{E}\left[X_\theta\left(\frac{1+\beta^2}{\beta^2 \cdot P + \lfloor M \cdot \theta \rceil}, 0\right)\right] \overset{(2.7)}{=} \frac{1+\beta^2}{\beta^2 \cdot P + \lfloor M \cdot \theta \rceil} \cdot \mathbb{E}\left[\mathrm{TP}_\theta\right] \\ &= \frac{\lfloor M \cdot \theta \rceil \cdot P \cdot (1+\beta^2)}{M \cdot (\beta^2 \cdot P + \lfloor M \cdot \theta \rceil)} = \frac{(1+\beta^2)\cdot P \cdot \theta^*}{\beta^2 \cdot P + M \cdot \theta^*}. \end{aligned} \tag{2.14}$$

Baseline: To determine θ that yields the highest expectation of $\mathrm{F}_\theta^{(\beta)}$, and therefore the DD baseline, we require the derivative of this metric. Let us say $f : [0,1] \to [0,1]$ is the continuous version of this metric and defined as follows:

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

First note that $\mathbb{E}\left[\mathrm{F}_\theta^{(\beta)}\right] = f(\lfloor M \cdot \theta \rceil / M)$. The derivative of this function is

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

It is strictly positive for all t in its domain; thus, f is strictly increasing in t. This means $\mathbb{E}\left[\mathrm{F}_\theta^{(\beta)}\right]$ given in Equation (2.14) 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,

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

Consequently, the optimization value $\theta_{\max}$ for the highest expectation is

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

Following this reasoning, the discrete form $\theta^*_{\max}$ is given by

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

2.6.10 Youden's J Statistic

The *Youden's J statistic* (J_θ), also called *Youden's index* or *(bookmaker) informedness*, was introduced by Youden [53] to capture the performance of a diagnostic test as a single statistic. It incorporates both TPR_θ and TNR_θ, discussed in Sections 2.6.5 and 2.6.6, respectively. J_θ shows how well the model can correctly predict both the positive as well as the negative observations. It is a metric that is naturally maximized.

Definition and Distribution: The J_θ is commonly defined as follows:

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

By using Equations (2.10) and (2.11), which provide the definitions of TPR_θ and TNR_θ in terms of TP_θ, the definition of J_θ can be reformulated as

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

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

$$J_\theta = X_\theta\left(\frac{M}{P\,(M-P)}, -\frac{\lfloor M\cdot\theta\rceil}{M-P}\right) \sim f_{X_\theta}\left(\frac{M}{P\,(M-P)}, -\frac{\lfloor M\cdot\theta\rceil}{M-P}\right),$$

with range

$$J_\theta \overset{(2.6)}{\in} \mathcal{R}\left(X_\theta\left(\frac{M}{P\,(M-P)}, -\frac{\lfloor M\cdot\theta\rceil}{M-P}\right)\right).$$

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

$$\begin{aligned}\mathbb{E}\left[J_\theta\right] &= \mathbb{E}\left[X_\theta\left(\frac{M}{P\,(M-P)}, -\frac{\lfloor M\cdot\theta\rceil}{M-P}\right)\right]\\ &\overset{(2.7)}{=} \frac{M}{P\,(M-P)}\cdot\mathbb{E}\left[\text{TP}_\theta\right] - \frac{\lfloor M\cdot\theta\rceil}{M-P} = 0.\end{aligned}$$

Baseline: The baseline of the J_θ is determined by searching for the DD classifier yielding the highest expectation, as J_θ is a metric that is naturally maximized. It is given by

$$\max_{\theta\in[0,1]}\mathbb{E}\left[J_\theta\right] = 0.$$

The expected value does not depend on θ. Consequently, the optimization value θ_{max} can be any value in the domain of θ. Thus,

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

This also holds for the discrete form θ^*_{max} of the optimizer:

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

2.6.11 Markedness

The *markedness* (MK_θ), also called *DeltaP* [54], is a performance measure mostly used in linguistics and social sciences. It combines PPV_θ and NPV_θ. These two measures are discussed in Sections 2.6.7 and 2.6.8. MK_θ indicates how cautious the model is in predicting observations as positive and also how cautious it is in predicting them as negative. It is a metric that naturally is maximized.

Definition and Distribution: MK_θ is commonly defined as follows:

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

This definition of MK_θ can be reformulated in terms of TP_θ by using Equations (2.12) and (2.13):

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

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, 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

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

with range

$$\text{MK}_\theta \overset{(2.6)}{\in} \mathcal{R} \left(X_\theta \left(\frac{M}{\lfloor M \cdot \theta \rceil (M - \lfloor M \cdot \theta \rceil)}, -\frac{P}{M - \lfloor M \cdot \theta \rceil} \right) \right).$$

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 derived as follows:

$$\begin{aligned}\mathbb{E}\left[\text{MK}_\theta\right] &= \mathbb{E}\left[X_\theta \left(\frac{M}{\lfloor M \cdot \theta \rceil (M - \lfloor M \cdot \theta \rceil)}, -\frac{P}{M - \lfloor M \cdot \theta \rceil} \right) \right] \\ &\overset{(2.7)}{=} \frac{M}{\lfloor M \cdot \theta \rceil (M - \lfloor M \cdot \theta \rceil)} \cdot \mathbb{E}\left[\text{TP}_\theta\right] - \frac{P}{M - \lfloor M \cdot \theta \rceil} = 0.\end{aligned}$$

Baseline: The baseline of the MK_θ is determined by searching for the DD classifier yielding the highest expectation, as MK_θ is a metric that is naturally maximized. It is given by

$$\max_{\theta \in [1/(2M), 1-1/(2M))} \left(\mathbb{E}\left[\text{MK}_\theta\right]\right) = 0.$$

The expected value does not depend on θ. Therefore, the optimization value $\theta_{\max}$ is in the set of allowed values for θ

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

This also means its discrete form $\theta^*_{\max}$ of the optimizer is in the set of the allowed discrete values

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

2.6.12 Accuracy

The *accuracy* (Acc_θ) is a performance measure that assesses how good a model is in correctly predicting the observations without distinguishing between positive or negative observations.

Definition and Distribution: Acc_θ is commonly defined as follows:

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

By using Equation (2.4), this can be restated as

$$\text{Acc}_\theta = \frac{2 \cdot \text{TP}_\theta + M - P - \lfloor M \cdot \theta \rceil}{M}. \tag{2.15}$$

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

$$\text{Acc}_\theta = X_\theta \left(\frac{2}{M}, \frac{M - P - \lfloor M \cdot \theta \rceil}{M} \right) \sim f_{X_\theta} \left(\frac{2}{M}, \frac{M - P - \lfloor M \cdot \theta \rceil}{M} \right),$$

with range

$$\text{Acc}_\theta \overset{(2.6)}{\in} \mathcal{R} \left(X_\theta \left(\frac{2}{M}, \frac{M - P - \lfloor M \cdot \theta \rceil}{M} \right) \right).$$

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

$$\begin{aligned} \mathbb{E}\left[\text{Acc}_\theta\right] &= \mathbb{E}\left[X_\theta \left(\frac{2}{M}, \frac{M - P - \lfloor M \cdot \theta \rceil}{M} \right) \right] \\ &\overset{(2.7)}{=} \frac{2}{M} \cdot \mathbb{E}\left[\text{TP}_\theta\right] + \frac{M - P - \lfloor M \cdot \theta \rceil}{M} \\ &= \frac{\left(M - \lfloor M \cdot \theta \rceil\right)\left(M - P\right) + \lfloor M \cdot \theta \rceil \cdot P}{M^2} \\ &= \frac{\left(1 - \theta^*\right)\left(M - P\right) + \theta^* \cdot P}{M}. \end{aligned} \tag{2.16}$$

Baseline: The baseline of the Acc_θ is determined by searching for the DD classifier yielding the highest expectation, as Acc_θ is a metric that is naturally maximized. Let us say that $f : [0, 1] \to [0, 1]$ is the continuous variant of the Acc_θ and given as

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

First, note that $\mathbb{E}\left[\text{Acc}_\theta\right] = f(\lfloor M \cdot \theta \rceil / M)$. The derivative of this function is given by

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

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 $\mathbb{E}[\text{Acc}_\theta]$ given in Equation (2.16). This is because the term $\lfloor M \cdot \theta \rceil / M$ is non-decreasing in θ. Hence,

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

This means that the optimization value $\theta_{\max} \in [0,1]$ is

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

Consequently, the discrete version $\theta^*_{\max} \in \Theta^*$ of the optimizer is given by

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

2.6.13 Balanced Accuracy

The *balanced accuracy* (BAcc_θ) is the mean of TPR_θ and TNR_θ, which are discussed in Sections 2.6.5 and 2.6.6. 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 metric BAcc_θ is commonly defined as follows:

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

By using Equations (2.10) and (2.11), this can be reformulated as

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

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

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

with range

$$\mathrm{BAcc}_\theta \overset{(2.6)}{\in} \mathcal{R}\left(X_\theta\left(\frac{M}{2P\,(M-P)}, \frac{M-P-\lfloor M\cdot\theta\rceil}{2\,(M-P)}\right)\right).$$

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:

$$\begin{aligned}\mathbb{E}\left[\mathrm{BAcc}_\theta\right] &= \mathbb{E}\left[X_\theta\left(\frac{M}{2P\,(M-P)}, \frac{M-P-\lfloor M\cdot\theta\rceil}{2\,(M-P)}\right)\right] \\ &\overset{(2.7)}{=} \frac{M}{2P\,(M-P)}\cdot\mathbb{E}\left[\mathrm{TP}_\theta\right] + \frac{M-P-\lfloor M\cdot\theta\rceil}{2\,(M-P)} = \frac{1}{2}.\end{aligned}$$

Baseline: The baseline of the BAcc_θ is determined by searching for the DD classifier yielding the highest expectation, as BAcc_θ is a metric that is naturally maximized. Since the expectation is constant, its baseline is

$$\max_{\theta\in[0,1]} \mathbb{E}\left[\mathrm{BAcc}_\theta\right] = \frac{1}{2}.$$

This means that the optimization value $\theta_{\max} \in [0,1]$ is

$$\theta_{\max} \in \arg\max_{\theta\in[0,1]} \mathbb{E}\left[\mathrm{BAcc}_\theta\right] = [0,1].$$

Consequently, the discrete version $\theta^*_{\max} \in \Theta^*$ is given by

$$\theta^*_{\max} \in \arg\max_{\theta^*\in\Theta^*} \mathbb{E}\left[\mathrm{Acc}_{\theta^*}\right] = \Theta^*.$$

2.6.14 Matthews' Correlation Coefficient

The *Matthews' correlation coefficient* (MCC_θ) was established by Matthews [55]. However, its definition is identical to that of the *Yule phi coefficient*, which was introduced by Yule [56]. 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 $[0,1]$. The MCC_θ is naturally maximized.

Definition and Distribution: The metric MCC_θ is commonly defined as follows:

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

By using Equations (2.2) and (2.4), this definition can be reformulated as

$$\mathrm{MCC}_\theta = \frac{M\cdot\mathrm{TP}_\theta - P\cdot\lfloor M\cdot\theta\rceil}{\sqrt{\lfloor M\cdot\theta\rceil\cdot P\,(M-P)\,(M-\lfloor M\cdot\theta\rceil)}}. \tag{2.17}$$

To ensure the denominator is non-zero, 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 (2.17) 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 (2.17):

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

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

$$\begin{aligned} \text{MCC}_\theta &= X_\theta \left(\frac{M}{C(M, P, \theta)}, \frac{-P \cdot \lfloor M \cdot \theta \rceil}{C(M, P, \theta)} \right) \\ &\sim f_{X_\theta} \left(\frac{M}{C(M, P, \theta)}, \frac{-P \cdot \lfloor M \cdot \theta \rceil}{C(M, P, \theta)} \right), \end{aligned}$$

with range

$$\text{MCC}_\theta \overset{(2.6)}{\in} \mathcal{R} \left(X_\theta \left(\frac{M}{C(M, P, \theta)}, \frac{-P \cdot \lfloor M \cdot \theta \rceil}{C(M, P, \theta)} \right) \right).$$

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 (2.7):

$$\begin{aligned} \mathbb{E}\left[\text{MCC}_\theta\right] &= \mathbb{E}\left[X_\theta \left(\frac{M}{C(M, P, \theta)}, \frac{-P \cdot \lfloor M \cdot \theta \rceil}{C(M, P, \theta)} \right) \right] \\ &\overset{(2.7)}{=} \frac{M}{C(M, P, \theta)} \cdot \mathbb{E}\left[\text{TP}_\theta\right] - \frac{P \cdot \lfloor M \cdot \theta \rceil}{C(M, P, \theta)} = 0. \end{aligned}$$

Baseline: The baseline of the MCC_θ is determined by searching for the DD classifier yielding the highest expectation, as MCC_θ is a metric that is naturally maximized. Since the expectation is constant, the expectation is also zero:

$$\max_{\theta \in [1/(2M), 1-1/(2M))} \mathbb{E}\left[\text{MCC}_\theta\right] = 0.$$

This means that the optimization value θ_{max} is

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

Consequently, the discrete version θ^*_{max} is

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

2.6.15 Cohen's Kappa

Cohen's kappa (κ_θ) is a less straightforward performance measure than the other measures discussed in this research. It is used to quantify the inter-rater reliability for two raters of categorical observations [57]. In our case, we compare the first rater, which is the DD classifier, with the perfect rater, which assigns the true label to each observation.

Definition and Distribution: Although there are several definitions for κ_θ, here we selected the following:

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

with $P_o^\theta = \text{Acc}_\theta$ as defined in Section 2.6.12 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:

$$P_o^\theta = \text{Acc}_\theta = \frac{\text{TP}_\theta + \text{TN}_\theta}{M},$$

and

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

By using Equations (2.15), (2.1), (2.2), (2.3) and (2.4) the above can be rewritten as

$$P_o^\theta = \frac{2 \cdot \text{TP}_\theta + M - P - \lfloor M \cdot \theta \rceil}{M},$$

and

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

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

$$\frac{\lfloor M \cdot \theta \rceil}{M} \neq \frac{P}{2P - M}. \tag{2.18}$$

The left-hand side is by definition in the interval $[0, 1]$. For the right-hand side to be in that interval, we first need $P/(2P - M) \geq 0$. Since $P \geq 0$, that means $2P - M > 0$; 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 (2.18) becomes

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

To summarize, 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 κ_θ, we obtain

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

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

$$a_{\kappa_\theta} = \frac{2M}{P\left(M - \lfloor M \cdot \theta \rceil\right) + (M - P)\lfloor M \cdot \theta \rceil},$$

and

$$b_{\kappa_\theta} = -\frac{2 \cdot \lfloor M \cdot \theta \rceil \cdot P}{P\left(M - \lfloor M \cdot \theta \rceil\right) + (M - P)\lfloor M \cdot \theta \rceil}.$$

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

$$\kappa_\theta = X_\theta\left(a_{\kappa_\theta}, b_{\kappa_\theta}\right) \sim f_{X_\theta}\left(a_{\kappa_\theta}, b_{\kappa_\theta}\right),$$

with range

$$\kappa_\theta \overset{(2.6)}{\in} \mathcal{R}\left(X_\theta\left(a_{\kappa_\theta}, b_{\kappa_\theta}\right)\right).$$

Expectation: As κ_θ is linear in TP_θ, its expectation can be derived:

$$\mathbb{E}\left[\kappa_\theta\right] = \mathbb{E}\left[X_\theta\left(a_{\kappa_\theta}, b_{\kappa_\theta}\right)\right] \overset{(2.7)}{=} a_{\kappa_\theta} \cdot \mathbb{E}\left[\text{TP}_\theta\right] + b_{\kappa_\theta} = 0.$$

Baseline: The baseline of the κ_θ is determined by searching for the DD classifier yielding the highest expectation, as κ_θ is a metric that is naturally maximized. Since the expectation is constant, the expectation is also zero. Thus,

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

This means that the optimization value $\theta_{\max}$ is in the set of all allowed values:

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

Consequently, the discrete version $\theta^*_{\max}$ is given by

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

2.6.16 Fowlkes-Mallows Index

The *Fowlkes-Mallows index* (FM_θ), also called *G-mean 1*, was introduced by Fowlkes and Mallows [58] as a way to calculate the similarity between two clusterings. It is the geometric average between TPR_θ and PPV_θ, which are discussed in Sections 2.6.5 and 2.6.7, respectively. It balances correctly predicting the actual positive observations (TPR_θ) and being cautious in predicting observations as positive (PPV_θ). This metric is naturally maximized.

Definition and Distribution: The metric FM_θ is commonly defined as follows:

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

By using the definitions of TPR_θ and PPV_θ in terms of TP_θ in Equations (2.10) and (2.12), respectively, 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

$$\text{FM}_\theta = X_\theta \left(\frac{1}{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}}, 0 \right) \sim f_{X_\theta} \left(\frac{1}{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}}, 0 \right),$$

with range

$$\text{FM}_\theta \overset{(2.6)}{\in} \mathcal{R} \left(X_\theta \left(\frac{1}{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}}, 0 \right) \right).$$

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

$$\begin{aligned} \mathbb{E}\left[\text{FM}_\theta\right] = \mathbb{E}\left[X_\theta \left(\frac{1}{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}}, 0 \right) \right] &\overset{(2.7)}{=} \frac{1}{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}} \cdot \mathbb{E}\left[\text{TP}_\theta\right] + 0 \\ &= \frac{\sqrt{P \cdot \lfloor M \cdot \theta \rceil}}{M} = \sqrt{\frac{\theta^* \cdot P}{M}}. \end{aligned}$$

Baseline: The baseline of the FM_θ is determined by searching for the DD classifier yielding the highest expectation, as FM_θ is a metric that is naturally maximized. The baseline is given by

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

because the expectation is a non-decreasing function in θ. Consequently, the optimizer $\theta_{\max}$ is given by

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

Consequently, the discrete version $\theta^*_{\max}$ is given by

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

2.6.17 G-mean 2

The *G-mean 2* ($\mathrm{G}_\theta^{(2)}$) was established by Kubat *et al.* [59]. This performance measure is the geometric average between the TPR_θ and TNR_θ, which we discuss in Sections 2.6.5 and 2.6.6, respectively. Hence, it balances correctly predicting the positive observations and correctly predicting the negative observations. It is a metric that is naturally maximized.

Definition and Distribution: The metric $\mathrm{G}_\theta^{(2)}$ is defined as follows:

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

Since TPR_θ needs the assumption $P>0$ and TNR_θ needs $N := M-P>0$, we have these restrictions also for $\mathrm{G}_\theta^{(2)}$. Consequently, $M>1$. Now, by using the definitions of TPR_θ and TNR_θ in terms of TP_θ in, respectively, Equations (2.10) and (2.11), we obtain

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

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

Expectation: Since $\mathrm{G}_\theta^{(2)}$ is not linear in TP_θ, we cannot easily use the expectation of TP_θ to determine that for $\mathrm{G}_\theta^{(2)}$. There are two statistics we can derive. The first is the expectation of a DD classifier labeling randomly $M-P$ observations positive ($\theta^* = \frac{M-P}{M}$). The expectation of this DD classifier is given by

$$\mathbb{E}\left[\mathrm{G}^{(2)}_{\theta^*=\frac{M-P}{M}}\right] = \frac{1}{\sqrt{P\cdot(M-P)}}\cdot\mathbb{E}\left[\mathrm{TP}_\theta\right] = \frac{\sqrt{P\cdot(M-P)}}{M}.$$

Secondly, we can determine the second moment:

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

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

Baseline: While we cannot derive the expectation of the $\mathrm{G}_\theta^{(2)}$, we can show its upper bounded by 0.5 by taking the inequality of the geometric mean and the arithmetic mean:

$$\begin{aligned}\mathbb{E}\left[\mathrm{G}_\theta^{(2)}\right] &= \mathbb{E}\left[\sqrt{\frac{\mathrm{TP}_\theta}{P}\cdot\frac{\mathrm{TN}_\theta}{N}}\right]\leq\frac{1}{2}\cdot\mathbb{E}\left[\frac{\mathrm{TP}_\theta}{P}+\frac{\mathrm{TN}_\theta}{N}\right]\\ &= \frac{1}{2}\cdot\left(\mathbb{E}\left[\frac{\mathrm{TP}_\theta}{P}\right]+\mathbb{E}\left[\frac{\mathrm{TN}_\theta}{N}\right]\right)=\frac{1}{2}\cdot(\theta^*+(1-\theta^*))=\frac{1}{2}.\end{aligned}$$

In the scenario that $P = M - P$, it can be observed that the DD classifier $\theta^* = \frac{M-P}{M}$ results in this upper bound. This implies that the maximum expectation can be achieved by randomly predicting 50% instances positive/negative on a perfectly balanced dataset.

2.6.18 Prevalence Threshold

A relatively new performance measure named *prevalence threshold* (PT_θ) was introduced by Balayla [46]. 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 of determining all statistics.

Definition and Distribution: The metric PT_θ is commonly defined as follows:

$$\mathrm{PT}_\theta = \frac{\sqrt{\mathrm{TPR}_\theta\cdot(1-\mathrm{TNR}_\theta)}-1+\mathrm{TNR}_\theta}{\mathrm{TPR}_\theta-1+\mathrm{TNR}_\theta}.$$

By using the definitions of TPR_θ and TNR_θ in terms of TP_θ (see Equations (2.10) and (2.11)), we obtain

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

It is clear that this performance measure is not a linear function of TP_θ, therefore we cannot easily calculate its expectation. However, there are more fundamental problems with PT_θ.

Division by Zero: The definition of PT_θ under the DD 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, as the denominator is zero. Furthermore, also the numerator is zero in that case. 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 optimization 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 PT_θ.

2.6.19 Threat Score / Critical Success Index

The *threat score* (TS_θ) [60], also called the *critical success index* [61] is a performance measure that is used for evaluation of forecasting binary weather events: it happens in a specific location or it does not. It was already used in 1884 to evaluate the prediction of tornadoes [61]. The TS_θ 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 metric TS_θ is defined as follows:

$$\mathrm{TS}_\theta = \frac{\mathrm{TP}_\theta}{\mathrm{TP}_\theta + \mathrm{FP}_\theta + \mathrm{FN}_\theta}.$$

By using Equations (2.2) and (2.3), 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 $X_\theta\,(a, b)$.

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

$$\mathbb{E}\left[\mathrm{TS}_\theta\right] = \sum_{k \in \mathcal{D}(\mathrm{TP}_\theta)} \frac{k}{P + \lfloor M \cdot \theta \rceil - k} \cdot \mathbb{P}(\mathrm{TP}_\theta = k).$$

Unfortunately, we cannot explicitly solve this sum, but it can be calculated numerically.

Baseline: To determine the maximum $\mathbb{E}\left[\mathrm{TS}_\theta\right]$ and the corresponding parameter $\theta_{\max}$, we first derive an upper bound for the expectation and show that this value is attained for a specific interval. We then prove that there is no θ outside this interval also yielding this value. Let us 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,

$$\begin{aligned}
\mathbb{E}\left[\mathrm{TS}_\theta\right] &= \sum_{k \in \mathcal{D}(\mathrm{TP}_\theta)} \frac{k}{P + \lfloor M \cdot \theta \rceil - k} \cdot \mathbb{P}(\mathrm{TP}_\theta = k) \\
&\leq \sum_{k \in \mathcal{D}(\mathrm{TP}_\theta)} \frac{k}{P + \lfloor M \cdot \theta \rceil - P} \cdot \mathbb{P}(\mathrm{TP}_\theta = k) \\
&= \frac{1}{\lfloor M \cdot \theta \rceil} \sum_{k \in \mathcal{D}(\mathrm{TP}_\theta)} k \cdot \mathbb{P}(\mathrm{TP}_\theta = k) = \frac{\mathbb{E}\left[\mathrm{TP}_\theta\right]}{\lfloor M \cdot \theta \rceil} \overset{(2.7)}{=} \frac{P}{M}.
\end{aligned}$$

Next, let $\theta_{\max} \in [1 - 1/(2M), 1]$, then we get

$$\begin{aligned}
\mathbb{E}\left[\mathrm{TS}_{\theta_{\max}}\right] &= \sum_{k=M-(M-P)}^{P} \frac{k}{P + M - k} \cdot \mathbb{P}(\mathrm{TP}_{\theta_{\max}} = k) \\
&= \frac{P}{P + M - P} \cdot \mathbb{P}(\mathrm{TP}_{\theta_{\max}} = P) = \frac{P}{M},
\end{aligned}$$

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 maximizer. Let us distinguish two scenarios for which we derived the baseline. In the scenario of $P = 1$, let us show that the interval of maximizers is actually $\theta_{\max} \in [1/(2M), 1]$. Let us say that $\theta \in [1/(2M), 1 - 1/(2M))$, then $0 < \lfloor M \cdot \theta \rceil < M$. This results in

$$\begin{aligned}
\mathbb{E}\left[\mathrm{TS}_\theta\right] &= \sum_{k=\max\{0, \lfloor M \cdot \theta \rceil - (M-1)\}}^{\min\{1, \lfloor M \cdot \theta \rceil\}} \frac{k}{1 + \lfloor M \cdot \theta \rceil - k} \cdot \mathbb{P}(\mathrm{TP}_\theta = k) \\
&= \frac{0}{1 + \lfloor M \cdot \theta \rceil - 0} \cdot \mathbb{P}(\mathrm{TP}_\theta = 0) + \frac{1}{1 + \lfloor M \cdot \theta \rceil - 1} \cdot \mathbb{P}(\mathrm{TP}_\theta = 1) \\
&= \frac{1}{\lfloor M \cdot \theta \rceil} \cdot \mathbb{P}(\mathrm{TP}_\theta = 1) = \frac{1}{\lfloor M \cdot \theta \rceil} \cdot \left(\frac{\binom{1}{1}\binom{M-1}{\lfloor M \cdot \theta \rceil - 1}}{\binom{M}{\lfloor M \cdot \theta \rceil}} \right) = \frac{1}{M},
\end{aligned}$$

which is exactly the upper bound $\mathbb{E}\left[\mathrm{TS}_{\theta_{\max}}\right] = P/M$ for $P = 1$. Let us now look at the scenario of $P > 1$. We show that the maximizers is in $\theta_{\max} \in [1 - 1/(2M), 1]$. 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

$$
\begin{aligned}
\mathbb{E}\left[\mathrm{TS}_{\theta'}\right] &= \sum_{k \in \mathcal{D}(\mathrm{TP}_{\theta'})} \frac{k}{P + \lfloor M \cdot \theta' \rceil - k} \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k) \\
&= \frac{k'}{P + \lfloor M \cdot \theta' \rceil - k'} \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k') \\
&+ \sum_{k \in \mathcal{D}(\mathrm{TP}_{\theta'}) \setminus \{k'\}} \frac{k}{P + \lfloor M \cdot \theta' \rceil - k} \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k) \\
&\leq \frac{k'}{P + \lfloor M \cdot \theta' \rceil - (P-1)} \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k') + \\
&\sum_{k \in \mathcal{D}(\mathrm{TP}_{\theta'}) \setminus \{k'\}} \frac{k}{P + \lfloor M \cdot \theta' \rceil - P} \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k) \\
&= \frac{k'}{\lfloor M \cdot \theta' \rceil + 1} \mathbb{P}(\mathrm{TP}_{\theta'} = k') + \sum_{k \in \mathcal{D}(\mathrm{TP}_{\theta'}) \setminus \{k'\}} \frac{k}{\lfloor M \cdot \theta' \rceil} \mathbb{P}(\mathrm{TP}_{\theta'} = k) \\
&< \frac{k'}{\lfloor M \cdot \theta' \rceil} \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k') + \sum_{k \in \mathcal{D}(\mathrm{TP}_{\theta'}) \setminus \{k'\}} \frac{k}{\lfloor M \cdot \theta' \rceil} \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k) \\
&= \frac{1}{\lfloor M \cdot \theta' \rceil} \sum_{k \in \mathcal{D}(\mathrm{TP}_{\theta'})} k \cdot \mathbb{P}(\mathrm{TP}_{\theta'} = k) = \frac{P}{M}.
\end{aligned}
$$

Hence, there is a strict inequality $\mathbb{E}\left[\mathrm{TS}_{\theta'}\right] < \frac{P}{M}$ and this means θ' is not a maximizer of the expectation. Consequently, the maximizers are only in the interval $[1 - 1/(2M), 1]$ for $P > 1$. In summary,

$$
\max_{\theta \in [0,1]} \mathbb{E}\left[\mathrm{TS}_{\theta}\right] = \frac{P}{M},
$$

and

$$
\theta_{\max} \in \arg\max_{\theta \in [0,1]} \mathbb{E}\left[\mathrm{TS}_{\theta}\right] = \begin{cases} \left[\frac{1}{2M}, 1\right] & \text{if } P = 1, \\ \left[1 - \frac{1}{2M}, 1\right] & \text{if } P > 1. \end{cases}
$$

Since $\theta^*_{\max}$ is the discretization of $\theta_{\max}$, we obtain

$$
\theta^*_{\max} \in \arg\max_{\theta^* \in \Theta^*} \mathbb{E}\left[\mathrm{TS}_{\theta^*}\right] = \begin{cases} \Theta^* \setminus \{0\} & \text{if } P = 1, \\ \{1\} & \text{if } P > 1. \end{cases}
$$

3 The Optimal Input-Independent Baseline for Binary Classification: The Dutch Draw

Contents

Based on [25]:
J. Pries, E.P. van de Bijl, J.G. Klein, S. Bhulai, and R.D. van der Mei, "The optimal input-independent baseline for binary classification: The Dutch Draw", *Statistica Neerlandica*, volume 77, number 4, pages 543-554, 2023.

Abstract

Before any binary classification model is taken into practice, it is essential to validate its performance on a proper test set. Without a frame of reference given by a baseline method, it is impossible to determine if a score is 'good' or 'bad'. The goal in this chapter is to examine all baseline methods that are independent of feature values and determine which model is the 'best' and why. By identifying which baseline models are optimal, a crucial selection decision in the evaluation process is simplified. We prove that the proposed *Dutch Draw* classifier resulting in the Dutch Draw baseline is the best *input-independent* classifier (independent of feature values) for all *order-invariant* measures (independent of sequence order) assuming that the samples are randomly shuffled. This means that the Dutch Draw baseline is the optimal baseline under these intuitive requirements and should therefore be used in practice.

3.1 Introduction

A *binary classification* model tries to answer the following question: "Should the instance be labeled as zero or one?" This question might seem simple, but there are many practical applications for binary classification, ranging from predicting confirmed COVID-19 cases [62], detecting malicious intrusions [63] to determining if a runner is fatigued or not [64]. Whenever a classification model is developed for a practical application, it is important to validate the performance on a test set.

A baseline is necessary to put the achieved performance in perspective. Without this frame of reference, only partial conclusions can be drawn from the results. An accuracy of 0.9 indicates that 90% of all predictions are correct. However, it could be that the model actually did not learn anything and such high accuracy could already be achieved by predicting only zeros. To put the performance in perspective, it should therefore be compared with some meaningful benchmark method, preferably with a state-of-the-art model.

Nevertheless, many state-of-the-art methods are instance-specific. They can rapidly change and often involve many fine-tuned parameters. Thus, as a necessary additional check in the development process, we plead in Chapter 2 for a supplementary baseline that is *general*, *simple*, and *informative*. This is used to test if the new model truly performs better than a simple model. It should be considered a major warning sign when a model is outperformed by, for example, a weighted coin flip. The model can use information about the feature values of a sample, yet it is outperformed by a model that does not even consider these values. Is the model then actually learning something productive?

A theoretical approach for binary classification, called the *Dutch Draw* (DD), is proposed in Chapter 2 based on *DD classifiers*. Such a classifier draws uniformly at random (u.a.r.) a subset out of all samples, and labels these 1, and the rest 0. The size of the drawn subset is optimized to obtain the optimal expected performance, which is the *DD baseline*. For most commonly used performance measures, a closed-form expression is given in Chapter 2.

There are infinitely many ways to devise a baseline method. We only investigate prediction models that do not take any information from the features into account, as this will result in a more general and simple baseline. We call these models *input-independent*. Irrespective of the input, the way that such a model predicts remains the same. Any newly developed model should at least beat the performance of these kinds of models, as an *input-independent* model cannot exploit patterns in the data to predict the labels more accurately. However, sometimes a model can get lucky by accidentally predicting the labels perfectly for a specific order of the

labels. The order of the samples should not influence the 'optimality' of a model. This is why we introduce the notion of *average-permutation-optimality*. Furthermore, the order of the samples should not change the outcome of the performance measure (*order-invariant*). This is not a strict condition, as most commonly used measures have this property. Under these restrictions, we prove that the DD baseline is *average-permutation-optimal* out of all *input-independent* classifiers for any *order-invariant* measure.

Our contributions are as follows: (1) we determine natural requirements for a general, informative, and simple baseline, and (2) we prove that the DD baseline is the optimal baseline under these requirements. These contributions improve the evaluation process of any new binary classification method.

The remainder of this chapter is organized as follows. First, the necessary preliminaries and notations are discussed in Section 3.2. Next, in Section 3.3 we determine requirements for a general, simple and informative baseline. Furthermore, we formally define what optimality entails under these requirements. In Section 3.4, an alternative definition for the *DD classifiers* is given, which is necessary for the main proof. In Section 3.5, we prove that the DD baseline is optimal. Finally, Section 3.6 summarizes the general findings and discusses possible future research opportunities.

3.2 Preliminaries

Next, we introduce some concepts and notations to lay the foundation for the main proof. *Binary classifiers* (Section 3.2.1) and *performance measures* (Section 3.2.2) for binary classification are discussed, which will play a crucial role in the proof of the main result.

3.2.1 Binary Classifiers

To find a good baseline for a *binary classification model*, we first have to discuss what a *binary classifier* actually is. To this end, let $\mathcal{X}$ be the feature space (think e.g., $\mathbb{R}^d$). Normally, a binary classifier is defined as a function $h : \mathcal{X} \times \mathbb{R} \to \{0, 1\}$ that maps feature values to zero or one, where the second input is used to model stochasticity. However, this classifier only classifies one sample at a time. Instead, we are interested in classifiers that classify *multiple* samples *simultaneously*: $h_M : \mathcal{X}^M \times \mathbb{R} \to \{0, 1\}^M$, where $M \in \mathbb{N} \setminus \{0\}$ denotes the number of samples that are classified. This gives classifiers the ability to predict k out of M samples positive precisely. Note that a *single sample* classifier h can simply be extended to classify M samples *simultaneously* by applying the classifier for each sample individually

using $h_M : ((x_1, \ldots, x_M), r) \mapsto (h(x_1, r), \ldots, h(x_M, r))$. Note that $r \in \mathbb{R}$ can be viewed as a random seed. Let $\mathcal{H}_M$ be the set of *all* binary classifiers that classify M samples at the same time.

3.2.2 Performance Measures for Binary Classification

To assess the effectiveness of a binary classification model, it is necessary to choose a *performance measure*, which quantifies how much the *predicted* labels agree with the *actual* labels. Namely, each sample indexed by i has feature values $\mathbf{x}_i \in \mathcal{X}$ and a corresponding label $y_i \in \{0, 1\}$. Let $\mathbf{X} := (\mathbf{x}_1 \ldots \mathbf{x}_M) \in \mathcal{X}^M$ be the combined feature values of M samples. Furthermore, let $\mathbf{Y} = (y_1, \ldots, y_M)$ denote the corresponding labels. A performance measure for binary classification is then defined as $\mu : \{0, 1\}^M \times \{0, 1\}^M \to \mathbb{R}$, where the first entry of μ is the predictions made by the classifier and the second entry is the corresponding labels. The performance of classifier h_M can now be written as: $\mu(h_M(\mathbf{X}, r), \mathbf{Y})$.

3.2.3 Undefined Cases

Some measures are undefined for specific combinations of $h_M(\mathbf{X}, r)$ and $\mathbf{Y}$. Take for example the *true positive rate* [65], which is the number of correctly *predicted* positives divided by the total number of *actual* positives. When there are no actual positives, the measure is ill-defined, as it divides by zero. Less obvious, the measure *negative predictive value* [65] is undefined when no negatives are predicted, as it is defined as the number of correctly predicted negatives divided by the total number of predicted negatives. Assigning a constant value C to undefined cases solves many issues. However, this can make it desirable for a classifier to always predict labels that lead to a previously undefined measure in order to minimize the measure. Therefore, we redefine μ from now on for every $\hat{\mathbf{Y}}, \mathbf{Y} \in \{0, 1\}^M$ to be equal to a specific constant C_{undef}, when $\mu(\hat{\mathbf{Y}}, \mathbf{Y})$ was undefined. We make a distinction for each objective (maximizing/minimizing). Let

$$C_{\text{undef}} := \begin{cases} \max_{\hat{\mathbf{Y}}, \mathbf{Y} \in \{0,1\}^M} \left\{ \mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right\} & \text{if minimizing,} \\ \min_{\hat{\mathbf{Y}}, \mathbf{Y} \in \{0,1\}^M} \left\{ \mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right\} & \text{if maximizing.} \end{cases}$$

It is therefore always disadvantageous for a classifier to predict a previously undefined case. By defining C_{undef} in this way, we do not have to omit such classifiers from our analysis.

3.3 Essential Conditions

To prove that the optimal DD classifier yields the 'optimal' baseline, we first have to define 'optimality'. When is a baseline considered to be optimal? To determine this, the following two questions must be answered: (1) which methods do we compare, and (2) how do we compare them? To this end, we define the notion of *input-independent* classifiers, *order-invariant* measures, and *average-permutation-optimality*.

3.3.1 Input-Independent Classifier

Any binary classifier can be used as a baseline. However, any good standardized baseline should be *general*, *simple*, and *informative*, as described in Chapter 2. Thus, it needs to be applicable to any domain, quick to train and clearly still beatable. To this end, we investigate all models that do not take any feature values into account, as they meet these three requirements. Without considering feature values, they can be applied to any domain. Furthermore, they do not require any training, because they cannot learn the relationship between the feature values and the corresponding labels. This makes them also clearly still beatable, as any newly developed model should leverage the information from the feature values to make better predictions.

A binary classifier $h_M \in \mathcal{H}_M$ is called *input-independent* if for all feature values $\mathbf{X}_i, \mathbf{X}_j \in \mathcal{X}^M$ and $r \in \mathbb{R}$ it holds that $h_M(\mathbf{X}_i, r) = h_M(\mathbf{X}_j, r) =: h_M(r)$, where the notation of $h_M(r)$ is chosen to visualize that the classifier h_M is not dependent on the input. An example of an input-independent classifier is a *coin flip*, as the feature values have no influence on the probability distribution of the coin. Let $\mathcal{H}_M^{i.i.} = \{h_M \in \mathcal{H}_M : h_M \text{ is input-independent}\}$ be the set of all input-independent binary classifiers. A newly developed model, that was optimized using the same performance measure, should always beat the performance of an *input-independent* model, as it gains information from the feature values. Otherwise, the model was not able to exploit this extra information to make better predictions.

3.3.2 Order-Invariant Measure

To assess the performance of a method, a measure needs to be chosen. Reasonably, the order of the samples should not change the outcome of this measure. If a measure has this property, we call it *order-invariant*. To define this formally, we have to introduce the following notions. Let S_M denote the set of all permutations

of a set of size M by

$$S_M := \left\{\pi : \{1,\dots,M\} \to \{1,\dots,M\} \text{ s.t. } \{\pi(i)\}_{i=1}^{M} = \{1,\dots,M\}\right\}.$$

To apply permutations to a matrix, we consider *sample-wise permutations*. For every $M \times K$ dimensional matrix $X = (\mathbf{x}_1 \dots \mathbf{x}_M)$, let X_π denote the sample-wise permutation under π. Thus, $X_\pi := \left(\mathbf{x}_{\pi(1)} \dots \mathbf{x}_{\pi(M)}\right)$, with $K \in \mathbb{N} \setminus \{0\}$ the number of features. This means that the matrix X is reordered by row.

A measure μ is *order-invariant* if for every permutation $\pi \in S_M$ and for all $h_M(\mathbf{X}, r), \mathbf{Y} \in \{0,1\}^M$ it holds that

$$\mu\left(h_M(\mathbf{X}, r), \mathbf{Y}\right) = \mu(h_M(\mathbf{X}, r)_\pi, \mathbf{Y}_\pi). \tag{3.1}$$

This means that any reordering of the coupled *predicted* and *actual* labels does not affect the performance score. This is not a hard restriction, as most measures have this property. Note for example that the number of *true positives* (TP), *true negatives* (TN), *false positives* (FP), and *false negatives* (FN) are all *order-invariant*. Most commonly used measures are a function of these four measures [66], making them also *order-invariant*.

3.3.3 Defining Optimality

To find the 'optimal' baseline, it is first essential to specify what 'optimality' entails. There are three important factors to discuss. First, a binary classifier can be *stochastic*, which is why we examine the *expected* performance. Second, the 'optimal' classifier is the best classifier with respect to the group of classifiers that is considered, denoted by $\tilde{\mathcal{H}}_M$. Finally, it is the 'best' for a *specific* dataset, but we consider all permutations of the dataset. Otherwise, the 'optimal' classifier would simply be the deterministic classifier that produces the exact labels accidentally. This phenomenon is similar to a broken clock that gives the correct time twice a day, but should not be used to determine the time.

With this in mind, we introduce the notion of *average-permutation-optimality*. A classifier is *average-permutation-optimal* if it minimizes/maximizes the expected performance for a random permutation of the test set out of all considered binary classifiers ($\tilde{\mathcal{H}}_M$). Thus,

$$h_M^{\min} \in \underset{h_M \in \tilde{\mathcal{H}}_M}{\arg\min} \left\{\mathbb{E}_{\pi \sim \mathcal{U}(S_M)}\left[\mathbb{E}_{r \in \mathbb{R}}\left[\mu(h_M(\mathbf{X}_\pi, r), \mathbf{Y}_\pi)\right]\right]\right\}, \tag{3.2}$$

and

$$h_M^{\max} \in \underset{h_M \in \tilde{\mathcal{H}}_M}{\arg\max} \left\{\mathbb{E}_{\pi \sim \mathcal{U}(S_M)}\left[\mathbb{E}_{r \in \mathbb{R}}\left[\mu(h_M(\mathbf{X}_\pi, r), \mathbf{Y}_\pi)\right]\right]\right\}. \tag{3.3}$$

3.4 Dutch Draw

A DD classifier is defined in Chapter 2 for $\theta \in [0, 1]$, as

$$\sigma_\theta(\mathbf{X}, \cdot) := (\mathbf{1}_E(i))_{i \in \{1,\dots M\}} \text{ with } E \subseteq \{1, \dots M\} \text{ drawn u.a.r. such that } |E| = \lfloor M \cdot \theta \rceil. \tag{3.4}$$

In other words, the classifier draws u.a.r. a subset E of size $\lfloor M \cdot \theta \rceil$ out of all samples, which it then labels as 1, while the rest is labeled 0. In this section, we introduce an alternative definition, that is used in the main proof, and show that all DD classifiers are *input-independent*.

3.4.1 Alternative Definition

Instead of the definition in Equation (3.4), we introduce an alternative definition for the DD classifiers to simplify the proof of the main result. Given a binary vector $(y_1, \dots, y_M) \in \{0, 1\}^M$ of length M, note that the number of ones it contains can be counted by taking the sum $\sum_{i=1}^{M} y_i$. Next, we define sets of binary vectors (of the same length) that contain the same number of ones. For all $j \in \{0, \dots, M\}$, define

$$\mathcal{Y}_j := \left\{ \hat{\mathbf{Y}} = (y_1, \dots, y_M) \in \{0, 1\}^M \text{ s.t. } \sum_{i=1}^{M} y_i = j \right\}. \tag{3.5}$$

In other words, the set $\mathcal{Y}_j$ contains all binary vectors of length M with exactly j ones and $M - j$ zeros.

A DD classifier selects u.a.r. E out of M samples and labels these as one, and the rest zero. Note that this is the same as taking u.a.r. a vector from $\mathcal{Y}_E$. To simplify notation, let $\mathcal{U}(A)$ denote the uniform distribution over a finite set A. Thus, when $X \sim \mathcal{U}(A)$ it must hold that $\mathbb{P}(X = a) = \frac{1}{|A|}$ for each $a \in A$. Now, a DD classifier σ_θ can be rewritten as

$$\sigma_\theta(\mathbf{X}, \cdot) := \hat{\mathbf{Y}} \text{ with } \hat{\mathbf{Y}} \sim \mathcal{U}\left(\mathcal{Y}_{\lfloor M \cdot \theta \rceil}\right). \tag{3.6}$$

Put differently, a DD classifier σ_θ chooses u.a.r. a vector with exactly $\lfloor M \cdot \theta \rceil$ ones as prediction out of all vectors with $\lfloor M \cdot \theta \rceil$ ones ($\mathcal{Y}_{\lfloor M \cdot \theta \rceil}$). This alternative definition simplifies the proof of the main result.

3.4.2 Input-Independence

Next, we discuss why all DD classifiers are *input-independent* (see Section 3.3.1). Note that a DD classifier σ_θ is independent of feature values, as it is only dependent on θ and M, see Equation (3.6). In other words, any DD classifier is by

definition *input-independent*. Instead of $\sigma_\theta(\mathbf{X}, r)$, we can therefore write $\sigma_\theta(r)$. To conclude, for every $\theta \in [0,1]$ it holds that $\sigma_\theta \in \mathcal{H}_M^{i.i.}$, which is the set of all input-independent binary classifiers.

3.4.3 Optimal Dutch Draw Classifier

The optimal DD classifier $\sigma_{\theta_{\text{opt}}}$ is determined by minimizing/maximizing the expected performance for the parameter θ out of all allowed parameter values Θ, as described in Chapter 2. Note that some measures are undefined for certain predictions, thus Θ is not always equal to $[0,1]$. Take e.g., the measure *precision* [65], which is defined as the number of true positives divided by the total number of predicted positives. Therefore, if no positives are predicted, the measure becomes undefined (division by zero). By adapting each measure according to Section 3.2.3, all undefined cases are resolved and $\Theta = [0,1]$ always holds.

Using the alternative definition of the DD classifier (see Equation (3.6)), we obtain

$$\theta^*_{\text{min}} \in \underset{\theta\in[0,1]}{\arg\min} \left\{ \mathbb{E}_{\hat{\mathbf{Y}}\sim\mathcal{U}(\mathcal{Y}_{\lfloor M\cdot\theta\rceil})} \left[\mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right] \right\},$$

and

$$\theta^*_{\text{max}} \in \underset{\theta\in[0,1]}{\arg\max} \left\{ \mathbb{E}_{\hat{\mathbf{Y}}\sim\mathcal{U}(\mathcal{Y}_{\lfloor M\cdot\theta\rceil})} \left[\mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right] \right\}. \tag{3.7}$$

Depending on the objective, either $\sigma_{\theta^*_{\text{min}}}$ or $\sigma_{\theta^*_{\text{max}}}$ is an optimal DD classifier. Observe that the optimal DD classifier depends on Y, thus a different dataset could lead to a different optimal DD classifier.

3.5 Theorem and Proof

After defining *input-independence* (Section 3.3.1), *order-invariance* (Section 3.3.2), *average-permutation-optimality* (Section 3.3.3), and introducing an alternative formulation for the DD classifier, all ingredients for the following theorem are present.

Theorem 1. *The optimal DD classifier $\sigma_{\theta_{opt}}$ is average-permutation-optimal out of all input-independent classifiers ($\mathcal{H}_M^{i.i.}$), for any order-invariant measure μ. In other words:*

$$\sigma_{\theta^*_{min}} \in \underset{h_M\in\mathcal{H}_M^{i.i.}}{\arg\min} \left\{ \mathbb{E}_{\pi\sim\mathcal{U}(S_M)} \left[\mathbb{E}_{r\in\mathbb{R}} \left[\mu(h_M(\mathbf{X}_\pi, r), \mathbf{Y}_\pi) \right] \right] \right\}, \tag{3.8}$$

and

$$\sigma_{\theta^*_{max}} \in \underset{h_M\in\mathcal{H}_M^{i.i.}}{\arg\max} \left\{ \mathbb{E}_{\pi\sim\mathcal{U}(S_M)} \left[\mathbb{E}_{r\in\mathbb{R}} \left[\mu(h_M(\mathbf{X}_\pi, r), \mathbf{Y}_\pi) \right] \right] \right\}. \tag{3.9}$$

Intuition Behind Proof: An *input-independent* classifier cannot learn the actual label from feature values. The predictions are therefore arbitrary. By averaging the performance over all permutations of the dataset (what we call *average-permutation-optimal*), it is only relevant how many labels are predicted to be zero (or one) by the classifier, as the performance measure is not dependent on the order (*order-invariance*). The optimal DD classifier is determined by optimizing the number of predicted zeros and ones, which makes this baseline *average-permutation-optimal*.

Proof. Let $h_M \in \mathcal{H}_M^{i.i.}$ be an *input-independent* classifier and let μ be a *order-invariant* measure, where we assume that μ needs to be maximized. The classifier is *average-permutation-optimal* if it maximizes the expected performance under a random permutation of the test set out of all *input-independent* classifiers (see Equation (3.3)). For any *input-independent* classifier h_M, it holds that

$$\mathbb{E}_{r\in\mathbb{R}}\left[\mu(h_M(\mathbf{X}_\pi, r), \mathbf{Y}_\pi)\right] = \mathbb{E}_{r\in\mathbb{R}}\left[\mu(h_M(r), \mathbf{Y}_\pi)\right]. \tag{3.10}$$

The input $\mathbf{X}_\pi$ is not relevant for the classification, and can thus be omitted.

In total, there are 2^M unique possible predictions in $\{0,1\}^M$. Denote these distinct vectors by $\hat{\mathbf{Y}}_1, \ldots, \hat{\mathbf{Y}}_{2^M}$ such that $\bigcup_{i=1}^{2^M} \hat{\mathbf{Y}}_i = \{0,1\}^M$. Next, the expectation in Equation (3.10) can be written out by:

$$\mathbb{E}_{r\in\mathbb{R}}\left[\mu(h_M(r), \mathbf{Y}_\pi)\right] = \sum_{i=1}^{2^M} \mathbb{P}(h_M(\cdot) = \hat{\mathbf{Y}}_i) \cdot \mu(\hat{\mathbf{Y}}_i, \mathbf{Y}_\pi). \tag{3.11}$$

As we need to prove *average-permutation-optimality*, we have to take the expectation of Equation (3.11) over all permutations. Using linearity of expectation gives the following:

$$\begin{aligned}\mathbb{E}_{\pi\sim\mathcal{U}(S_M)}&\left[\sum_{i=1}^{2^M} \mathbb{P}(h_M(\cdot) = \hat{\mathbf{Y}}_i) \cdot \mu(\hat{\mathbf{Y}}_i, \mathbf{Y}_\pi)\right] \\ &= \sum_{i=1}^{2^M} \mathbb{P}(h_M(\cdot) = \hat{\mathbf{Y}}_i) \cdot \mathbb{E}_{\pi\sim\mathcal{U}(S_M)}\left[\mu(\hat{\mathbf{Y}}_i, \mathbf{Y}_\pi)\right].\end{aligned} \tag{3.12}$$

Instead of taking the expectation of a sum, we now take the sum of expectations.

The measure μ is *order-invariant*, thus using Equation (3.1) gives the following:

$$\mu(\hat{\mathbf{Y}}_i, \mathbf{Y}_\pi) = \mu((\hat{\mathbf{Y}}_i)_{\pi^{-1}}, (\mathbf{Y}_\pi)_{\pi^{-1}}) = \mu((\hat{\mathbf{Y}}_i)_{\pi^{-1}}, \mathbf{Y}). \tag{3.13}$$

Applying a permutation does not change a *order-invariant* measure μ. In this case, we apply the inverse permutation π^{-1} to retrieve $\mathbf{Y}$. Because of Equation (3.13), it therefore also holds that

$$\mathbb{E}_{\pi\sim\mathcal{U}(S_M)}\left[\mu(\hat{\mathbf{Y}}_i, \mathbf{Y}_\pi)\right] = \mathbb{E}_{\pi\sim\mathcal{U}(S_M)}\left[\mu((\hat{\mathbf{Y}}_i)_{\pi^{-1}}, \mathbf{Y})\right]. \tag{3.14}$$

S_M is a *group*, which is why the set of all *inverse permutations* is the same as the set of all *permutations* [67], [68]. Given that the permutations are drawn u.a.r., taking the expectation over all the *inverse permutations* is the same as taking the expectation over all *permutations*. When permutation π is drawn u.a.r., it namely holds that $\mathbb{P}(\pi = s) = \mathbb{P}(\pi = s^{-1}) = \frac{1}{|S_M|}$ for all $s \in S_M$. Therefore,

$$\begin{aligned}\mathbb{E}_{\pi\sim\mathcal{U}(S_M)}\left[\mu((\hat{\mathbf{Y}}_i)_{\pi^{-1}}, \mathbf{Y})\right] &= \sum_{s\in S_M}\left(\mu((\hat{\mathbf{Y}}_i)_{s^{-1}}, \mathbf{Y})\cdot\mathbb{P}(\pi = s)\right)\\ &= \sum_{s\in S_M}\left(\mu((\hat{\mathbf{Y}}_i)_{s^{-1}}, \mathbf{Y})\cdot\mathbb{P}(\pi = s^{-1})\right)\\ &= \mathbb{E}_{\pi\sim\mathcal{U}(S_M)}\left[\mu((\hat{\mathbf{Y}}_i)_{\pi}, \mathbf{Y})\right].\end{aligned} \tag{3.15}$$

Thus, π^{-1} can be replaced with π in Equation (3.14). Recall that $\mathcal{Y}_j$ is the set of all binary vectors of length M with j ones (see Equation (3.5)). Furthermore, note that applying a u.a.r. chosen permutation $\pi \in S_M$ on $\hat{\mathbf{Y}}_i \in \mathcal{Y}_j$ is the same as selecting u.a.r. $\hat{\mathbf{Y}} \in \mathcal{Y}_j$ as outcome, because for every $\hat{\mathbf{Y}}_\star \in \mathcal{Y}_j$ it holds that

$$\mathbb{P}\left(\hat{\mathbf{Y}} = \hat{\mathbf{Y}}_\star\right) = \frac{1}{|\mathcal{Y}_j|} \text{ with } \hat{\mathbf{Y}} \sim \mathcal{U}\left(\mathcal{Y}_j\right),$$

and

$$\mathbb{P}\left((\hat{\mathbf{Y}}_i)_\pi = \hat{\mathbf{Y}}_\star\right) = \frac{1}{|\mathcal{Y}_j|}, \text{ with } \pi \sim \mathcal{U}\left(S_M\right).$$

The latter follows, as for each $\hat{\mathbf{Y}}_\star, \hat{\mathbf{Y}}_\triangle \in \mathcal{Y}_j$ there are exactly as many permutations to go from $\hat{\mathbf{Y}}_i$ to $\hat{\mathbf{Y}}_\star$ as permutations to go from $\hat{\mathbf{Y}}_i$ to $\hat{\mathbf{Y}}_\triangle$.

Let $|\hat{\mathbf{Y}}_i|$ denote the number of ones in $\hat{\mathbf{Y}}_i$. Now, we can rewrite the expectation $\mathbb{E}_{\pi\sim\mathcal{U}(S_M)}[\cdot]$ over all permutations into an expectation over a u.a.r. drawn vector with the same number of ones, by

$$\mathbb{E}_{\pi\sim\mathcal{U}(S_M)}\left[\mu((\hat{\mathbf{Y}}_i)_\pi, \mathbf{Y})\right] = \mathbb{E}_{\hat{\mathbf{Y}}\sim\mathcal{U}\left(\mathcal{Y}_{|\hat{\mathbf{Y}}_i|}\right)}\left[\mu(\hat{\mathbf{Y}}, \mathbf{Y})\right]. \tag{3.16}$$

Using Equations (3.14), (3.15), and (3.16) in combination with Equation (3.12) gives

$$\sum_{i=1}^{2^M} \mathbb{P}(h_M(\cdot) = \hat{\mathbf{Y}}_i) \cdot \mathbb{E}_{\pi \sim \mathcal{U}(S_M)} \left[\mu(\hat{\mathbf{Y}}_i, \mathbf{Y}_\pi) \right] = \sum_{i=1}^{2^M} \mathbb{P}(h_M(\cdot) = \hat{\mathbf{Y}}_i) \cdot \mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}\left(\mathcal{Y}_{|\hat{\mathbf{Y}}_i|}\right)} \left[\mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right].$$

We have now eliminated all permutations from the equation. Note that the expectation in the right-hand side is the same for each $\hat{\mathbf{Y}}_i \in \mathcal{Y}_j$. In other words, the expectation is the same for two vectors, when they have the same number of ones. Grouping the vectors with the same number of ones, gives

$$\sum_{i=1}^{2^M} \mathbb{P}(h_M(\cdot) = \hat{\mathbf{Y}}_i) \cdot \mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}\left(\mathcal{Y}_{|\hat{\mathbf{Y}}_i|}\right)} \left[\mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right] = \sum_{j=0}^{M} \mathbb{P}(h_M(\cdot) \in \mathcal{Y}_j) \cdot \mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}(\mathcal{Y}_j)} \left[\mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right].$$

Instead of summing over all possible binary vectors $\hat{\mathbf{Y}}_i \in \{0,1\}^M$, all vectors with the same number of ones are grouped together, as they have the same expectation. All the probability mass of the grouped vectors is also added up. Note that it is thus only relevant for a classifier in which group $\mathcal{Y}_j$ the prediction $h_M(\cdot)$ belongs.

For any $j \in \{0, \ldots, M\}$ it holds that $\mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}(\mathcal{Y}_j)} \left[\mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right]$ is bounded by maximizing over all possible values of j. Thus,

$$\mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}(\mathcal{Y}_j)} \left[\mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right] \leq \max_{j' \in \{0,\ldots,M\}} \mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}(\mathcal{Y}_{j'})} \left[\mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right]. \tag{3.17}$$

Observe that $\sum_{j=0}^{M} \mathbb{P}(h_M(\cdot) \in \mathcal{Y}_j) = 1$ and $\mathbb{P}(h_M(\cdot) \in \mathcal{Y}_j) \geq 0$ hold for each j. Therefore, it follows using Equation (3.17) that

$$\sum_{j=0}^{M} \mathbb{P}(h_M(\cdot) \in \mathcal{Y}_j) \cdot \mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}(\mathcal{Y}_j)} \left[\mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right] \leq \max_{j' \in \{0,\ldots,M\}} \mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}(\mathcal{Y}_{j'})} \left[\mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right].$$

Consequently, we have found an upper bound for Equation (3.9). Namely,

$$\max_{h_M \in \mathcal{H}_M^{i.i.}} \left\{ \mathbb{E}_{\pi \sim \mathcal{U}(S_M)} \left[\mathbb{E}_{r \in \mathbb{R}} \left[\mu(h_M(\mathbf{X}_\pi, r), \mathbf{Y}_\pi) \right] \right] \right\} \leq \max_{j \in \{0,\ldots,M\}} \left\{ \mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}(\mathcal{Y}_j)} \mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right\}. \tag{3.18}$$

Equality in (3.18) holds for any classifier $h_M \in \mathcal{H}_M^{i.i.}$, when all probability mass is given to $\arg\max_{j\in\{0,\dots,M\}} \left\{\mathbb{E}_{\hat{\mathbf{Y}}\sim\mathcal{U}(\mathcal{Y}_j)}\mu(\hat{\mathbf{Y}},\mathbf{Y})\right\}$. In other words, the maximum can only be attained if

$$\sum_{j_{\max}\in\arg\max_{j\in\{0,\dots,M\}}\left\{\mathbb{E}_{\hat{\mathbf{Y}}\sim\mathcal{U}\left(\mathcal{Y}_j\right)}\mu(\hat{\mathbf{Y}},\mathbf{Y})\right\}} \mathbb{P}(h_M(\cdot)\in\mathcal{Y}_{j_{\max}}) = 1. \tag{3.19}$$

A classifier $h_M \in \mathcal{H}_M^{i.i.}$ can therefore only attain the maximum if all predictions belong to a group $\mathcal{Y}_j$ or possibly multiple groups that all maximize the expectation.

Remember that the DD selects the optimal classifier based on Equation (3.7), which leads to

$$\lfloor M\cdot\theta^*_{\max}\rceil \in \underset{j\in\{0,\dots,M\}}{\arg\max}\left\{\mathbb{E}_{\hat{\mathbf{Y}}\sim\mathcal{U}(\mathcal{Y}_j)}\left[\mu(\hat{\mathbf{Y}},\mathbf{Y})\right]\right\}.$$

Combining this with the alternative definition of the DD (Equation (3.6)) directly gives that

$$\sum_{j_{\max}\in\arg\max_{j\in\{0,\dots,M\}}\left\{\mathbb{E}_{\hat{\mathbf{Y}}\sim\mathcal{U}\left(\mathcal{Y}_j\right)}\mu(\hat{\mathbf{Y}},\mathbf{Y})\right\}} \mathbb{P}(\sigma_{\theta^*_{\max}}(\cdot)\in\mathcal{Y}_{j_{\max}}) = 1.$$

This shows in combination with Equation (3.19) that the optimal DD classifier *actually* attains the bound given in Equation (3.18). It follows that

$$\sigma_{\theta^*_{\max}} \in \underset{h_M\in\mathcal{H}_M^{i.i.}}{\arg\max}\left\{\mathbb{E}_{\pi\sim\mathcal{U}(S_M)}\left[\mathbb{E}_{r\in\mathbb{R}}\left[\mu(h_M(\mathbf{X}_\pi,r),\mathbf{Y}_\pi)\right]\right]\right\}.$$

Similarly, we also find that when μ needs to be *minimized*, it follows that

$$\sigma_{\theta^*_{\min}} \in \underset{h_M\in\mathcal{H}_M^{i.i.}}{\arg\min}\left\{\mathbb{E}_{\pi\sim\mathcal{U}(S_M)}\left[\mathbb{E}_{r\in\mathbb{R}}\left[\mu(h_M(\mathbf{X}_\pi,r),\mathbf{Y}_\pi)\right]\right]\right\}.$$

Thus, we conclude that the optimal DD classifier attains the minimum/maximum expected performance and is therefore *average-permutation-optimal* for all *input-independent* classifiers with a *order-invariant* measure. □

This means that the optimal DD classifier is the best general, simple, and informative baseline.

3.6 Discussion and Conclusion

A baseline is crucial to assess the performance of a prediction model. However, there are infinitely many ways to construct a baseline. As a necessary check in the development process, in Chapter 2 we plead for a supplementary baseline that is *general*, *simple*, and *informative*. In this chapter, we have therefore examined all baselines that are independent of feature values, which makes them general and relatively simple. Additionally, these baselines are also informative, as it should be considered a major warning sign when a newly developed model is outperformed by a model that does not take any feature values into account. In this chapter, we have shown that, out of all *input-independent* binary classifiers, the DD baseline is *average-permutation-optimal* for any *order-invariant* measure. Our findings improve the evaluation process of any new binary classification method, as we have proven that the DD baseline is ideal to gauge the performance score of a newly developed model.

Next, we discuss two points that could be considered an 'unfair' advantage for the DD baseline. First of all, we have considered in this chapter classifiers that predict M labels *simultaneously*. This gives classifiers a potential advantage over classifying each sample *sequentially*, as e.g., exactly k out of M samples can be labeled positive. This can only be done sequentially when a classifier is allowed to track previous predictions or to change based on the number of classifications it has made. Even with this advantage, we believe that all *input-independent* models still remain clearly beatable by a newly developed model.

Secondly, the DD baseline can be derived for most commonly used measures without any additional knowledge about the number of positive labels P. Nonetheless, it was shown in Chapter 2 that the DD baseline can only be calculated for the measure *accuracy* when it is known if $P \geq M/2$ holds. If the distribution of the training set is the same as the test set, the training set can be used to determine whether $P \geq M/2$ is likely to hold. Furthermore, a domain expert could estimate whether it is likely that a dataset contains more positives than negatives. Take for example a cybersecurity dataset, where there are often significantly fewer harmful instances and more normal instances [69]. There are thus many ways to estimate if $P \geq M/2$ holds. Nevertheless, even if the DD baseline uses this information (only for the *accuracy*), we believe that any newly developed model should still beat the DD baseline, as it does not use any feature values to improve prediction.

Finally, we address future research opportunities. In this chapter, we have only considered *binary* classification. A natural extension would be to also consider *multiclass* classification [70], probabilistic classification, or regression (between $[0, 1]$). Is a strategy similar to the DD optimal in these cases? Can a closed-

form expression of the optimal baseline be derived? We believe that the three introduced properties (namely, input-independent, order-invariant, and average-permutation-optimal) are still relevant for these problems. This could help identify what kind of classifier is considered to be optimal. In Chapter 2, we stated that the DD baseline could be used to scale existing measures. This chapter provides more motivation to scale measures with the DD baseline and not by using any other *input-independent* classifier. Yet, how each measure should be scaled in order to maximize the explainability behind a performance score could still be investigated. Most performance measures reduce to a linear function of TP, which makes a DD classifier that predicts as many positives/negatives (without making the performance measure undefined) optimal. However, there are non-linear cases, such as the *G-mean 2*, that lead to a non-trivial baseline. Take e.g., $M = 50$ and $P = 5$, which leads to a DD baseline of approximately 0.4877 with $\theta^*_{\text{max}} = 0.54$ (see Figure 3.1).

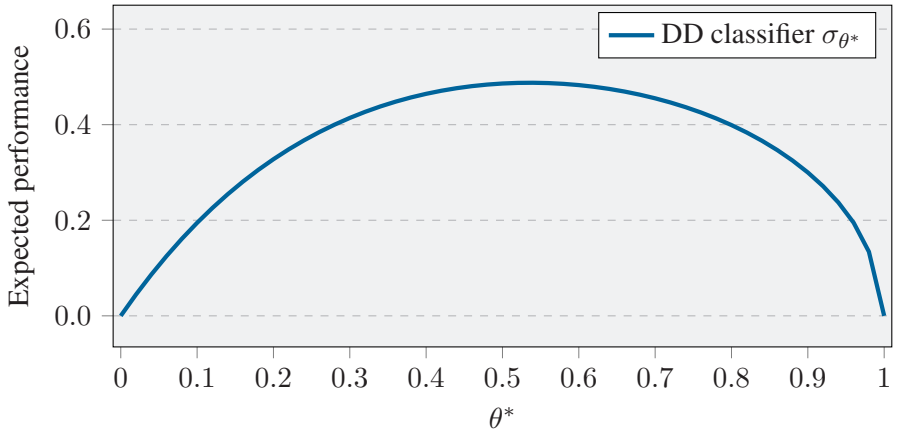
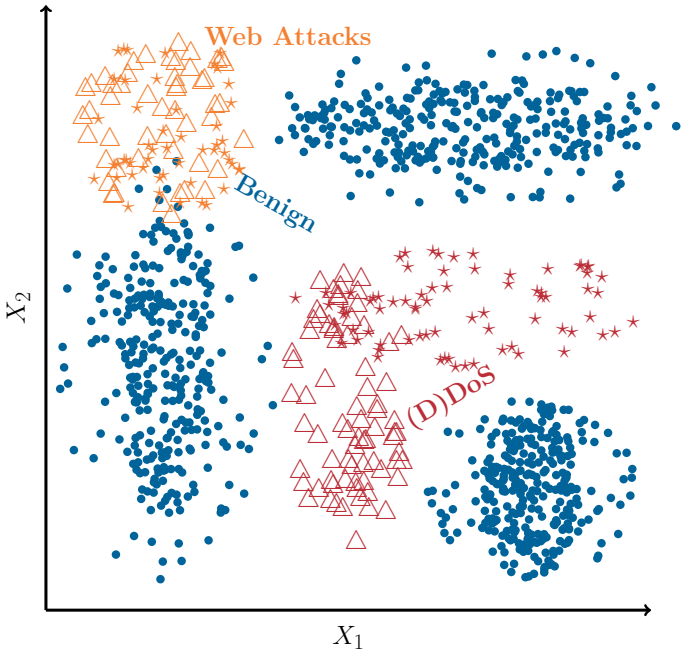


Figure 3.1: Expected DD classifier performance for G-mean 2 with $M = 50$ and $P = 5$

Non-linear cases could perhaps also lead to situations where multiple (not all) $\mathcal{Y}_j$ groups are optimal (see Equation (3.19)). We have not encountered this behavior for common performance measures, but it would be interesting to identify performance measures that do have this property.

Part II

Intrusion Detection



4 Detecting Novel Application Layer Cybervariants Using Supervised Learning

Contents

Based on [26]:
E.P. van de Bijl, J.G. Klein, J. Pries, R.D. van der Mei, and S. Bhulai, "Detecting novel application layer cybervariants using supervised learning", *International Journal on Advances in Security*, volume 15, number 3 & 4, pages 75-85, 2022.

Abstract

Cyberdefense mechanisms, such as *network intrusion detection systems*, predominantly use *signature-based* approaches to detect known malicious activities in network traffic effectively. Unfortunately, constructing a database with signatures is very time-consuming and this approach can only detect known cyberattacks. *Machine learning* algorithms are known to be effective software tools in detecting known intrusions, but whether they can detect unseen cyberattack variants has not been studied. In this chapter, we study to what extent binary classification models are accurately able to detect novel variants of *application layer* targeted cyberattacks. To be more precise, we focus on detecting two types of intrusion variants, namely *(distributed) denial-of-service* and *web* attacks, targeting the *Hypertext Transfer Protocol* of a web server. We demonstrate mathematically how two datasets are converted in three different experimental setups. The results of classification models deployed in these setups are benchmarked using the *Dutch Draw* baseline method, as discussed in Chapters 2 and 3 The contributions of this research are as follows: we provide a procedure to create intrusion detection datasets combining information from the *transport*, *network*, and *application* layer to be directly used for *machine learning* purposes. We show that specific variants are successfully detected by these classification models trained to distinguish benign connections from those of another variant. Despite this result, we show that classifier performance is asymmetric: a model trained on benign data and one attack, then tested on benign data and a different attack, does not necessarily yield a similar score when the training and testing sets are swapped. At last, we show that increasing the number of different variants in the training set does not necessarily lead to a higher detection rate of unseen variants. Selecting the right combination of a machine learning model with a (small) set of known intrusions included in the training data can result in a higher novel intrusion detection rate.

4.1 Introduction

Network security has become more challenging today as the Internet is used for virtually all information operations, such as storage and retrieval. The rat race between attackers and defenders is perpetual as new tools and techniques are continuously developed to attack web servers containing this information. Tremendous problems for organizations and individuals arise when legitimate users cannot access data due to cyberattacks. Modern attacks are designed to mimic legitimate user behavior and target vulnerabilities in *application-layer* protocols, such as the *Hypertext Transfer Protocol* (HTTP). This mix makes detecting them a challenging and complex task.

Defenders often use an *intrusion detection system* (IDS) to detect intrusions. An IDS can be viewed as a burglar alarm in the cybersecurity field [4]. It monitors network traffic to detect malicious activity and triggers an alarm when suspicious behavior is found. Generally speaking, these systems use two main types of methodologies: *signature-based* and *anomaly-based* [71]. A signature-based detector compares observed network events against patterns that correspond to known threats. In contrast, anomaly-based detectors search for malicious traffic by constructing a notion of normal behavior and flags activities that do not conform to this notion. Where signature-based is time-consuming but effective, anomaly-based often has a high false-positive rate. Within anomaly detection methods, *machine learning* (ML) algorithms are getting more attention as they might overcome this problem.

The thought of using ML algorithms to detect intrusions is not new. Various studies have been performed on the use of ML to detect cyberattacks. Unfortunately, there is a striking imbalance between the extensive amount of research on ML-based anomaly detection techniques for *intrusion detection* (ID) and the rather clear lack of operational deployments [6]. ML algorithms are highly flexible and are adaptive methods to find patterns in big stacks of data [7], but they seem better at this task than discovering meaningful outliers [6]. Modern cyberattacks often occur in large quantities and thus do not entirely conform to the premise that patterns cannot be found for these outliers. Therefore, using ML for the task of detecting these attacks is essential.

There appear to be two issues when looking at anomaly-based ML research in ID [8], [9]. First, the performance of most of these methods is measured on outdated datasets [10]. This makes it hard to estimate the performance of these methods on modern network traffic. A major issue is that benign and malicious traffic composition in these datasets does not represent modern real-time environments. Also, there used to be a lack of representative publicly available ID datasets, but

the cyberdefense community noticed this lack and recently more intrusion datasets have been generated [23]. Still, the available datasets are often limited to features extracted from the transport and network layer and lack application-layer features. Thus, not all attainable features are extracted in these datasets. Second, it is not examined how supervised learning methods detect novel variants of known attacks. The performance of these methods is measured in either a *closed-world learning setting* in which training and test classes are the same or an *open-world learning setting* with unrelated attacks. However, how the methods perform in an open-world setting with novel variants is not tested.

This chapter aims to study to what extent ML models are accurately able to detect novel variants of known cyberattacks. To be more precise, we use *supervised binary classifiers* to learn from a dataset containing benign and application-layer cyberattacks and we evaluate them on their ability to detect unseen variants of these attacks. We focus on two cyberattack variants: *denial-of-service* (DoS) and its distributed form (DDoS) and *web* attacks. We examine how the selected classifiers perform when using a single cyberattack in the training dataset on this task. Afterward, we study the effect of combining malicious variants in the training phase on the performance of classifiers detecting unseen variants. The results of this binary classification problem are benchmarked using the *Dutch Draw* (DD) baseline method, as discussed in Chapters 2 and 3. Furthermore, we provide a procedure to transform raw network traffic data into ML-usable datasets containing information from the network, transport, and application layer. The code of this procedure is publicly available [31].

The main contributions can be summarized as follows: First, we show that ML classifiers are to a great extent able to detect known cyberattacks in a closed-world setting when presented with sufficient data. Second, we show that there are situations where these classifiers can detect a novel variant when they are trained to detect a different variant. However, this is not a two-way street: learning to detect attack A and being able to also detect attack B does not imply th does not studyhe case. Third, we show that training on imbalanced data harms the evaluation performance of some ML classifiers. We have demonstrated that variants included in the CIC-IDS-2017 seem not identical to the same variants in the CIC-IDS-2018. Finally, we demonstrate that using many variants to detect a novel attack is unnecessary. Sometimes, a few known attacks can lead to the highest detection rate.

The organization of this chapter is as follows. Section 4.2 gives a literature overview regarding detecting novel intrusions with ML. Section 4.3 describes the selected datasets and how they are modified into ML-applicable datasets and states metadata about them. In addition, a set of ML models is given to conduct the experiments. Section 4.4 outlines the conducted experiments. Section 4.5 shows

the results of the conducted experiments. Finally, we conclude and summarize in Section 4.6.

4.2 Related Work

Detection of novel attacks with supervised learning techniques has been studied before in the context of *transfer learning* (TL). TL is an ML paradigm where a model trained on one task is used as a starting point for another task. Zhao *et al.* [72] introduces a feature-based TL approach to find novel cyberattacks by mapping source and target datasets in an optimized feature representation. This approach is, however, very dependent on a similarity parameter and the dimensions of the new feature space. Therefore, Zhao *et al.* [73] extend this method by proposing another approach to automatically find a relationship between the novel and known attacks. Both of these approaches are tested on an outdated dataset, and it does not contain variants of a single cyberattack. In this research, we are interested in detecting novel variants rather than known or unrelated ones. In Wu *et al.* [74], a *convolutional neural network* is also used to detect novel attacks in a TL setup. Still, in t study, it is not studied if learning one specific attack affects the detection of another novel variant. Our research's experiments resemble those performed in Taghiyarrenani *et al.* [75]. In their research, an ID method is proposed that transfers knowledge between networks by combining unrelated attacks to train on. More recent work focuses on applying *deep neural networks* in the context of TL for ID tasks [76].

4.3 Data

We discuss the procedure for converting raw network traffic into usable ID datasets containing information from the network, transport, and application layers for ML purposes. We detail the converted and extracted features to clarify which features are included. Furthermore, we provide metadata describing the final datasets. Finally, we give the classification models and their set of considered hyperparameters to detect novel variants.

4.3.1 Data Sources

A perfect ID dataset should at least be up-to-date, correctly labeled, publicly available, contain real network traffic with all kinds of attacks and normal user behavior, and span over a long time [23]. The main reasons for a lack of appropriate datasets satisfying these properties are: (1) privacy concerns regarding recording real-world network traffic, and (2) labeling being very time-consuming. However,

synthetic or anonymized datasets that satisfy some of these ideal properties have been generated. It is therefore recommended to test methodologies on multiple datasets instead of only one [6]. In this research, we focus on the detection of malicious variants. For that reason, we have selected the CIC-IDS-2017 [77] and the CIC-IDS-2018 [78] datasets created by the *Canadian Institute for Cybersecurity* (CIC). These datasets are correctly labeled, publicly available, up-to-date, and contain several malicious cyberattacks.

4.3.2 Feature Extraction

The CIC provides the selected datasets in two formats: a set of raw network traffic (`pcap`) files and a set of files containing extracted features by a network analysis tool called CICFlowMeter [79]. These features mainly describe network and transport protocol activities. However, there are no features describing application-layer activities. As this study focuses on detecting application-layer cyberattacks, it is desirable to have a dataset containing application-layer features. Therefore, we start with the raw internet traffic format and have selected a feature extraction tool matching this requirement.

The feature extraction tool used in this study is the open-source network traffic analyzer called Zeek (formerly Bro) [11]. Zeek is a passive standalone IDS that derives an extensive log set describing network activity. These logs include an exhaustive record of all sessions seen on the wire. Zeek was also used as a feature extraction tool for the creation of other popular network ID datasets, e.g., DARPA98 [80] from the *Defense Advanced Research Projects Agency* (DARPA) and the UNSW-NB15 [81] from the *University of New South Wales* (UNSW). Zeek has a good track record in creating ID datasets and therefore an appropriate tool.

By default, Zeek generates a large set of log files, but not all are required for this research. We limit ourselves to the *Transmission Control Protocol* (TCP) entries given in the connection logs (`conn.log`), describing network and transport layer activity, and HTTP interactions given in their corresponding logs (`http.log`). These log files include entries showing malicious activities. The entries in the connection log files are transport-layer sessions, while the HTTP log file consists of entry logs showing conversations between a client and a web server. Entries between these logs are unilaterally linked as each HTTP entry is assigned to a single connection entry. Malicious activities that are not (D)DoS or web attacks are excluded as we only focus on these attacks.

4.3.3 Feature Engineering

We describe how the extracted features are converted into ML-ready features. This section states the additional created features, which are replaced for better extraction of patterns, and how we smartly one-hot-encode categorical features. We start by describing the feature engineering steps in the connection log file, and afterward, we do the same for the HTTP log file.

Connection Log: Zeek counts the number of packets and bytes transferred in each connection. Table 4.1 shows additional created features from these counters. A higher-level statistic called the *producer-consumer ratio* (PCR) [82] shows the ratio between sending and receiving packets between the hosts. In a TCP connection, an originator host is an uploader if a PCR is close to 1.0 and purely a downloader if it is close to -1.0.

Table 4.1: Network layer engineered features

Feature	Description	Type
orig_bpp	$\frac{\text{orig\_bytes}}{\text{orig\_packets}}$	Float
resp_bpp	$\frac{\text{resp\_bytes}}{\text{resp\_packets}}$	Float
PCR	$\frac{\text{orig\_bytes}-\text{resp\_bytes}}{\text{orig\_bytes}+\text{resp\_bytes}}$	Float

The feature `conn_state` constructed by Zeek refers to the final state of a TCP connection. This state is determined by registering flags exchanged during the communication between hosts. Looking only at the end of a connection implies that the establishment and termination of the connection are merged. Preliminary results showed that classifiers were better able to find patterns in (D)DoS traffic when differentiating between establishing a connection and its termination. On this note, we replaced the `conn_state` feature with features describing both ends of a connection. The *3-way handshake* is the correct way to establish a TCP connection before data is allowed to be sent. This procedure is, however, not always correctly executed and incorrect establishments can indicate misuse. Hosts can terminate TCP connections gracefully, or not. A graceful termination occurs when both hosts send a packet with a *final* (FIN) flag. When a host sends a packet containing a *reset* (RST) flag, it will abruptly end a TCP connection, which is very common in practice. If neither is the case, connections are in theory still open. In Table 4.2, we distinguish different establishment and termination scenarios by looking at the exchanged flags between the hosts. Each of these scenarios is included in the data as a binary feature. Other Zeek connection log flags are one-hot-encoded for both the originator and responder.

Table 4.2: TCP connection establishment and termination scenarios

Feature	Description
S0	No SYN packet is observed
S1	Merely a connection attempt (SYN), but no reply
REJ1	A connection attempt but replied with a RST packet
S2	A connection attempt followed by SYN-ACK, but no final ACK
REJ2O	Scenario S2 but originator sends RST packet
REJ2R	Scenario S2 but responder sends RST packet
S3	Connection is established according to the 3-way handshake
WEIRD	A connection attempt but none of the above cases were observed
OPEN	A connection was established, but no FIN or RST flag is observed
TERM	Connection gracefully terminated by originator and receive
CLSO	Originator sends a FIN flag but receiver did not respond
CLSR	Receiver sends a FIN flag but originator did not respond
RSTO	Originator abruptly ends connection by sending an RST flag
RSTR	Receiver abruptly ends connection by sending an RST flag

HTTP Log: Communication in this protocol starts with a client sending a request message to a web server and this server will, hopefully, reply with a response message. Both message types consist of a start-line, zero or more header fields, an empty line indicating the end of the header fields, and possibly a message body. The start-line of a request message, called the request-line, contains three components: a method (command), a path to apply this command on, and an HTTP version indicating the version a client wants to use. Hosts must agree on the HTTP version to use before they continue talking. If they disagreed with the HTTP version, a '-1' is imputed to distinguish it from other versions.

The feature `method` indicates the command in the request message and shows the one-word command given in the request line. Commonly used commands are 'GET', 'HEAD', 'POST', 'PUT', 'DELETE', 'CONNECT', 'OPTIONS', 'TRACE', and 'PATCH', but other commands also exist. This categorical feature is one-hot-encoded to one of those common commands to limit the number of options. If an uncommon command is given, it will be assigned to a feature called `method_other`, while if no command is given at all, it is assigned to `method_-`.

A web server applies a `method` on the *Uniform Resource Identifier* (URI) stated in a request line. An *Uniform Resource Locator* (URL) is a part of the URI that identifies a resource and provides the means to locate it. This URL can be parsed

in different components by a library called *urllib* [83]. Figure 4.1 gives an example of how this tool splits a URL into four components. We extracted descriptive statistics from each component by counting the number of special characters (not letters or digits), the number of characters, and the number of unique characters. A typical URL constitutes three components: a path, a query, and a fragment. Statistics are extracted for each of those components. For example, one extracted feature called `URL_path_len` describes the length of the path of a URL. In addition, Zeek extracts `host` (only netloc), the `referrer` (all components), and these descriptive statistics are also extracted for these features.

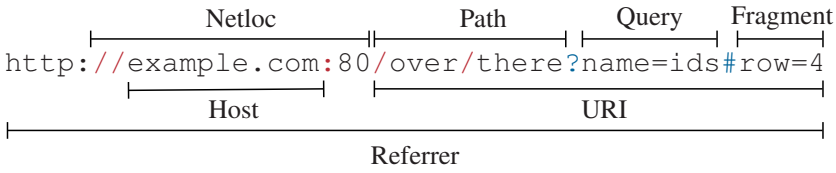


Figure 4.1: Example URL parsed by `urllib` showing four components (above) and coverage of the extracted features by Zeek (below)

Web servers process received request messages and reply to them with a response message. In the status line of this message is the agreed HTTP version stated and a response code if the web server can process the request. The response codes are grouped by their first digit. So, for example, the error code 404 is assigned to the `4xx` code. Furthermore, it registers the type of data (e.g., application, audio, example, font, image, model, text, or video) sent by the web server to the client or vice versa. This info is one-hot-encoded similarly as the `method` for both directions.

4.3.4 Final Dataset

The log files are merged into a single dataset after feature engineering them. The resulting dataset consists of HTTP interactions, while in contrast, the datasets provided by the CIC consist of connection flows. Connection log features are added to the HTTP entry features to combine application, network, and transport layer features. This merge gives a dataset with a total of 103 features. The CIC-IDS-2017 consists of 533,845 instances and the CIC-IDS-2018 has 9,595,037 instances.

Table 4.3 shows the distribution of the labels of the entries. The benign/malicious ratio is roughly balanced for the CIC-IDS-2017, while it is more imbalanced for the CIC-IDS-2018. We observe a clear imbalance between the malicious classes if we differentiate between cyberattacks. For example, the *HTTP Unbearable*

Load King (Hulk) attack generated a lot more HTTP entries in comparison to a *Slowloris* or *GoldenEye*. The same can be observed for web attacks. The *Brute Force* and *XSS* web attacks are more occurring in the dataset than the *SQL injection* attack.

Table 4.3: Class distribution of HTTP entries

		CIC-IDS-2017		CIC-IDS-2018	
Class	**Type**	**Amount**	**Percentage**	**Amount**	**Percentage**
Botnet	DDoS	736	00.28%	142,925	04.28%
GoldenEye	DoS	7,908	02.97%	27,345	00.82%
HOIC	DDoS	0	00.00%	1,074,379	32.15%
Hulk	DoS	158,513	59.48%	1,803,160	53.95%
LOIC	DDoS	95,683	35.90%	289,328	08.65%
SlowHTTPTest	DoS	1,416	00.53%	0	00.00%
Slowloris	DoS	2,245	00.84%	4,950	00.15%
		266,501	100.00%	3,342,807	100.00%
Brute Force	web	7,311	79.93%	13,144	54.02%
SQL Injection	web	12	00.13%	57	00.23%
XSS	web	1,824	19.94%	11,134	45.75%
		9,147	100.00%	24,335	100.00%
Benign	-	258,197	48.37%	6,252,950	65.00%
Malicious	-	275,648	51.63%	3,366,422	35.00%
		533,845	100.00%	9,619,372	100.00%

4.3.5 Models

We select four ML algorithms for our classification problem: *decision tree* (DT), *random forest* (RF), *k-nearest neighbors* (KNN), and *Gaussian naive Bayes* (GNB). A grid search approach is performed to find the optimal hyperparameters for these algorithms. Table 4.4 shows the considered parameters for each model. The optimal set of parameters for each model is used on the test dataset by selecting the highest F_1 score achieved on a validation set. As there was a limited amount of computational time, the hyperparameter space of computationally expensive models like KNN is smaller than simpler models like DT.

Table 4.4: Hyperparameters options for the selected classifiers

Model	Scikit Parameter	Options
GNB	var_smoothing	1e-200
DT	criterion	[Gini , Entropy]
	splitter	[Best, Random]
	class_weight	[None, Balanced]
	max_features	[Auto, None, Sqrt, log2]
RF	criterion	[Gini, Entropy]
	class_weight	[None, Balanced]
	max_features	Auto
	n_estimators	[10, 50, 100, 250]
KNN	n_neighbors	5
	algorithm	[Ball Tree, KD Tree]

4.4 Experimental Setup

In this section, we elaborate on the experiments that were conducted. We tested ML models to detect cyberattacks in three different experimental setups. Before we elaborate on those three setups, we start with mathematical preliminaries, how it is split into the train, validation, and test sets, and how we turned each experiment into a binary classification problem. After describing the experiments, we will discuss how the models are evaluated and tested against the DD.

4.4.1 Preliminaries

Suppose we have an ID dataset $\mathbf{X}$ consisting of M instances and K feature values each. Without loss of generality, we assume $\mathbf{X} \in \mathbb{R}^{M \times K}$. Let $\mathcal{M} := \{1, 2, \ldots, M\}$ denote the indices of the instances. Each instance i has a corresponding label y_i. Let $\mathbf{y} := (y_1, y_2, \ldots y_M)$ denote the vector containing all labels. The datasets consist of benign traffic and a set of malicious cyberattack variants. Therefore, let $\mathcal{L} := \{l_0, l_1, l_2, \ldots l_C\}$ be the set of possible values each instance y_i could have as label. Here, label l_0 is the *Benign* label, and $\{l_1, l_2 \ldots, l_C\}$ is the set of C different cyberattack types included in our data. Let $\mathcal{I}_k := \{i \in \mathcal{M} | y_i = l_k\}$ be the set of instances that have label l_k.

4.4.2 Train-Test Split

It is common practice in ML to split a dataset into two non-overlapping sets: a training and a test set. Binary classification models use the training set to learn a

relationship between the response and explanatory variables. Let us denote $\mathcal{M}_{\text{train}}$ as the instances that are assigned to the training dataset and $\mathcal{M}_{\text{test}}$ as the instances that are assigned to the test dataset. It should hold that $\mathcal{M}_{\text{train}}, \mathcal{M}_{\text{test}} \subset \mathcal{M}$ with the property that $\mathcal{M}_{\text{train}} \cap \mathcal{M}_{\text{test}} = \emptyset$, $|\mathcal{M}_{\text{train}}| + |\mathcal{M}_{\text{test}}| = M$ and we should select a ratio R such that $|\mathcal{M}_{\text{train}}| \cdot R = |\mathcal{M}_{\text{test}}|$. Typically, R is selected so that there is an 80:20 train/test ratio. As we investigate stochastic and deterministic prediction models, the train and test split procedure is repeated multiple times to get a proper average performance for these models.

4

There are multiple classes in the dataset, so we have added another requirement for the train-test split: we want to have a similar class distribution in both the training and test dataset. This is also known as *stratified sampling*. The train-test split of the instances should match the class distribution of the original dataset as closely as possible:

$$\frac{|\mathcal{M}_{\text{train}} \cap \mathcal{I}_k|}{|\mathcal{M}_{\text{train}}|} \approx \frac{|\mathcal{M}_{\text{test}} \cap \mathcal{I}_k|}{|\mathcal{M}_{\text{test}}|} \quad \forall k \in \{0, 1, 2, \ldots, C\}.$$

Furthermore, we require that at least two observations of each class be selected to allow hyperparameter tuning.

4.4.3 Hyperparameter Tuning

In Section 4.3.5, we discussed the considered hyperparameters for the selected ML classifiers. Selecting the right hyperparameters is vital for good performance. Hyperparameters are compared by taking the average performance of the ML models tested on different validation datasets. Train-validation splits are created in the same manner as the train-test split. The hyperparameters yielding the highest F_1 score are selected to test the models on the testing dataset.

4.4.4 Setups

The classification problem at hand is a multiclass classification problem when $C > 1$. We will, however, treat each problem as a binary classification problem by mapping all malicious classes towards a single label $l_{\text{malicious}}$ when $\mathbf{y}$ is presented to the model to train on and when evaluating. The classifiers are tested on these datasets in three different experimental setups.

Detecting Known Attacks: First, we study to what extent the selected classifiers can detect known attacks in a closed-world learning setting. The achieved detection rate could indicate an upper bound to the novel detection rate of the corresponding malicious class. To test this, the training data and the evaluation data contain the same two classes: *Benign* (l_0) and one malicious cyberattack (l_k).

More specifically, the training dataset for this first experiment for malicious attack $k \in \{1, 2, \ldots, C\}$, denoted by $\mathcal{T}_{1,k}$, consists of the following set instances:

$$\mathcal{T}_{1,k} = (\mathcal{I}_0 \cup \mathcal{I}_k) \cap \mathcal{M}_{\text{train}}.$$

The number 1 indicates that the set belongs to the first experiment. We evaluate the classifiers' performance on the following evaluation dataset in the first experimental setting, denoted by $\mathcal{E}_{1,k}$, using the same malicious attack (l_k):

$$\mathcal{E}_{1,k} = (\mathcal{I}_0 \cup \mathcal{I}_k) \cap \mathcal{M}_{\text{test}}.$$

For example, we let a model train to distinguish *Benign* from *Hulk* and test on the same two classes.

Detecting Novel Variants: Second, we examine to what degree classifiers can detect a novel variant when the training dataset contains benign traffic and one different variant. For example, we train a classifier to distinguish *Benign* from *Hulk* entries and evaluate the trained model on a test set containing *Benign* and *Low Orbit Ion Cannon* (LOIC). Both the *Hulk* and *LOIC* labels are converted to $l_{\text{malicious}}$, to keep the binary classification setting. This experiment shows us how similar the novel test attack is to the known training attack. Let us select $i, j \in \{1, 2, \ldots, C\}$ such that $i \neq j$. The training instances for this second experiment are derived as follows:

$$\mathcal{T}_{2,i} = (\mathcal{I}_0 \cup \mathcal{I}_i) \cap \mathcal{M}_{\text{train}}.$$

In contrast to the previous experiment, the included malicious instances are not from the same class and the evaluation dataset consists of:

$$\mathcal{E}_{2,j} = (\mathcal{I}_0 \cup \mathcal{I}_j) \cap \mathcal{M}_{\text{test}}.$$

If $i = j$, we would get the same datasets as in experimental setup 1.

Class Importance to Detect Novel Variants: Finally, we study what we call *class importance*: how crucial is including a variant in the training dataset on the novel cyberattack detection performance? Does learning on a combination of multiple attacks help identify novel variants? We look at combinations of cyberattacks in the training set and test the trained model on detecting a novel attack. For example, we train on *Benign* data combined with attacks like *LOIC* and *Hulk*, and test on a dataset containing *Benign* data along with a different, novel attack such as *SlowHTTPTest*. More formally, let us denote $E_{3,j}$ as the evaluation set containg malicious attack $j \in \{1, 2, \ldots, C\}$. The instances in this set are derived as follows:

$$E_{3,j} = (\mathcal{I}_0 \cup \mathcal{I}_j) \cap \mathcal{M}_{\text{test}}.$$

Now, we want to find which set of malicious variants leads to the highest novel cyberattack (l_j) detection rate. Say $\mathcal{P}(\mathcal{B}) \coloneqq \{\mathcal{A} \mid \mathcal{A} \subseteq \mathcal{B}\}$ is the powerset of set $\mathcal{B}$. So, the considered cyberattack combinations in the training dataset for novel attack l_j are derived as follows:

$$\mathcal{S}_j \coloneqq \mathcal{P}(\mathcal{L} \setminus \{l_0, l_j\}) \setminus \{\emptyset\}.$$

The empty set is excluded from this powerset as there needs to be at least one cyberattack to be included in the training dataset. Suppose we take now a random cyberattack set $s \in \mathcal{S}_j$. Let us define the set of instances having a label included in this set s as $\mathcal{I}_s \coloneqq \{i \in \mathcal{M} | y_i \in s\}$. Then the training dataset of this third experiment $T_{3,s}$, using set s cyberattack types, becomes:

$$\mathcal{T}_{3,s} = (\mathcal{I}_0 \cup \mathcal{I}_s) \cap \mathcal{M}_{\text{train}}.$$

With this formulation, we can study which set of known variants $s \in \mathcal{S}_j$ leads to the highest novel cyberattack l_j detection rate.

4.4.5 Evaluation Metrics

In our binary classification task, the *positive* class represents malicious instances while the *negative* class represents benign entries. Let us denote $y_i \in \{0, 1\}$ as the actual label of an instance i where '0' is the negative class (l_0) and '1' represents the positive class ($l_{\text{malicious}}$). A confusion matrix is constructed by comparing the binary predictions $\hat{y}$ of a classifier with the actual labels y. This 2×2 dimensional matrix contains four base measures: the number of *true positives* (TP), *true negatives* (TN), *false positives* (FP), and *false negatives* (FN). TP and TN show the number of correctly predicted instances, while FP and FN show the number of mistakes. These four measures form the basis of any binary evaluation metric.

The performance of a binary classification model is quantified by one or more evaluation metrics, which are a function of one or more base measures. The evaluation metric to test the selected classifiers is the F_1 score, which is the harmonic mean between *recall* and *precision*. Recall is the ratio of intrusions the classifiers successfully detected, while precision is the ratio between TP and the number of *positively predicted* instances. Hence:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

Taking the *harmonic* mean between those two gives the F_1 score, which is defined as:

$$F_1 = \frac{2}{\text{Recall}^{-1} + \text{Precision}^{-1}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}.$$

For cyberattacks aiming to exhaust a resource, it is better to have a low false alarm rate than a high recall as blocking all malicious traffic is unnecessary. We simply want to prevent the resource from being overloaded and prevent blocking legit HTTP requests. This makes this specific task different from detecting intrusions in general, as the cost of a false negative is higher. Still, optimizing only precision is not desirable. Therefore, the F_1 score is an appropriate middle ground as it optimizes the harmonic mean of those metrics. When data is imbalanced, this score is more suitable than accuracy as it corrects this imbalance.

4.4.6 Dutch Draw Baseline

Analyzing evaluation metric scores enables meaningful comparison of classifier performance. However, it does not indicate what the scores mean without some frame of reference. Baselines help interpret results as models can only be considered appropriate when outperforming them. Therefore, we have selected the DD baseline as this baseline method helps compare the performances of the selected ML models. This method gives a baseline, which is the score of the optimal input-independent random classifier. The selected evaluation metric to compare classifiers is the F_1 score and it follows from Chapter 2 that the corresponding DD baseline is given by $\frac{2P}{2P+M}$. To construct a baseline for a evaluation dataset E, the number of positives is given by $P = |E \setminus (\mathcal{I}_0 \cap \mathcal{M}_{\text{test}})|$ and $M = |\mathcal{M}_{\text{test}}|$.

4.5 Results

Now, we show the results of the three experimental setups performed in this research. The results of the experiments were gathered by testing the selected classifiers on 20 different train-test splits for the CIC-IDS-2017 and 10 different splits for the CIC-IDS-2018. Furthermore, as the CIC-IDS-2018 is very large and there was limited computational time, a subset of the data was used for hyperparameter tuning. For the DT and RF techniques, 10% was randomly selected for hyperparameter tuning. As the KNN model, with the selected hyperparameter options, is computationally very expensive, we were limited to only using 1% (randomly) of the training data for hyperparameter tuning. The same percentage of data was required in the training process to evaluate this model in a reasonable time. For the CIC-IDS-2017, no subset sampling was required for training purposes. In our experiments, we have performed multiple hold-out-cross validation splits with each split an 80/20 split randomly. Before splitting the data, all redundant features (features with only zero values) are removed as these features do not contain any new information. A validation set (20%) is randomly selected in the training set to obtain the best hyperparameters for each model.

4.5.1 Detecting Known Attacks

ML classifiers were tested to determine whether they could distinguish benign HTTP interactions from interactions labeled with a predefined known cyberattack by training and testing on the same classes. We start by discussing the results of detecting web attacks and show the results of detecting (D)DoS attacks. Table 4.5 shows the average F_1 scores for the selected ML models and the corresponding standard deviations. Each row corresponds to the selected cyberattack for the setup (l_k, where $k \in \{1, 2, \ldots, C\}$). The relatively lowest scores are highlighted in red. It can be observed that almost in all scenarios the classifiers outperform the DD baseline, for which we have taken the average expectation over all train-test splits. For the CIC-IDS-2018, we observe that the GNB and KNN model outperform the DD baseline, but the scores (highlighted in red) were relatively low. The models could not detect any of the *SQL injection* instances in all train-test splits. All other setups indicate that the ML models could find patterns to distinguish normal traffic from a malicious variant.

Table 4.5: F_1 scores of classifiers in detecting known web attacks

		DD baseline	GNB		DT		RF		KNN	
Dataset	**Attack**	**Exp**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
CIC-IDS-2017	Brute Force	0.0536	0.5711	0.0034	0.9994	0.0008	0.9994	0.0003	0.9983	0.0005
	SQL Injection	0.0001	0.9500	0.2236	0.8419	0.2690	0.8833	0.2484	0.0833	0.2059
	XSS	0.0139	0.9044	0.0065	0.9975	0.0033	0.9950	0.0022	0.9901	0.0043
CIC-IDS-2018	Brute Force	0.0042	0.8108	0.0059	0.9994	0.0003	0.9996	0.0002	0.9861	0.0018
	SQL Injection	0.0000	0.0134	0.0006	0.8834	0.0591	0.8655	0.0834	0.0000	0.0000
	XSS	0.0035	0.9977	0.0009	0.9998	0.0003	0.9999	0.0002	0.9927	0.0027

The same analysis is performed for (D)DoS attacks. Table 4.6 shows the average F_1 scores if classifiers were tested to detect known (D)DoS attacks. In almost all scenarios, the considered models could learn the relevant characteristics of the considered attacks. One exception, highlighted in red, is the GNB model trained and tested on the *SlowHTTPTest* attack. This model obtained a high recall (0.997), but a poor score on its precision (0.154). Even though the model can detect most malicious instances, many false positives existed.

4.5.2 Detecting Novel Attacks with One Attack Learned

Let us relax the closed-world assumption: What if our trained algorithm sees a different variant of the learned attack? Figure 4.2 shows the average F_1 scores achieved by the classifiers in this experiment when detecting web attacks. The diagonal of this matrix shows the F_1 scores of the closed-world assumption, also obtainable from Table 4.5, while the off-diagonal values were the scores of detecting novel attacks. We can observe that the GNB model can not detect all *Brute*

Table 4.6: F_1 scores of classifiers in detecting known (D)DoS attacks

		DD baseline	GNB		DT		RF		KNN	
Dataset	**Attack**	**Exp**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
CIC-IDS-2017	Botnet	0.0057	1.0000	0.0000	0.9971	0.0046	0.9998	0.0008	0.9909	0.0076
	GoldenEye	0.0577	0.9972	0.0010	0.9997	0.0002	1.0000	0.0000	0.9983	0.0006
	Hulk	0.5511	0.9990	0.0002	0.9999	0.0000	1.0000	0.0000	0.9999	0.0000
	LOIC	0.4257	0.9999	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000
	SlowHTTPTest	0.0108	0.2339	0.2065	0.9955	0.0042	0.9956	0.0031	0.9874	0.0046
	Slowloris	0.0171	0.9013	0.0078	0.9976	0.0016	0.9969	0.0023	0.9929	0.0035
CIC-IDS-2018	Botnet	0.0437	0.9998	0.0001	1.0000	0.0000	1.0000	0.0000	0.9974	0.0011
	GoldenEye	0.0087	0.9919	0.0006	0.9843	0.0010	0.9914	0.0004	0.9536	0.0051
	HOIC	0.2558	0.9964	0.0001	0.9964	0.0001	0.9964	0.0001	0.9961	0.0002
	Hulk	0.3658	0.9999	0.0000	1.0000	0.0000	1.0000	0.0000	0.9997	0.0000
	LOIC	0.0847	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000
	Slowloris	0.0016	0.9876	0.0018	0.9982	0.0012	0.9986	0.0007	0.9586	0.0054

Force attacks. Surprisingly, a high score is obtained when this model is not trained on the *Brute Force* attack but on the *XSS* attack.

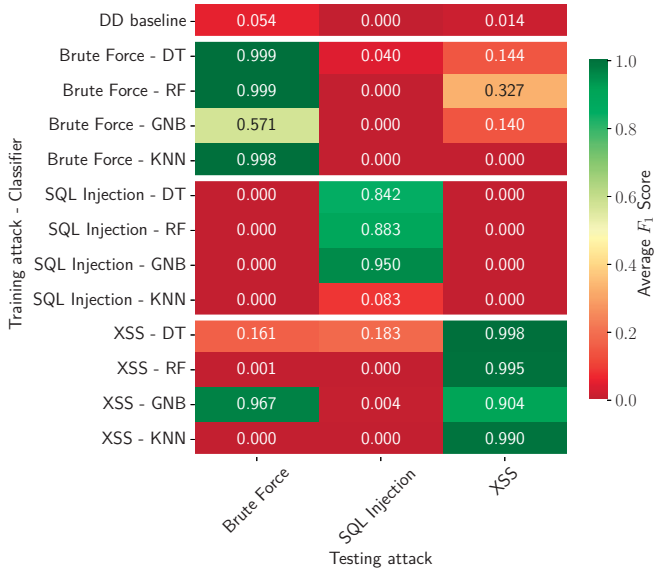


Figure 4.2: Classifier performance in detecting known and novel web attacks on the CIC-IDS-2017 dataset

If we look at the results extracted from the CIC-IDS-2018 dataset, we see a difference compared to the CIC-IDS-2017. Figure 4.3 shows that the DT model is very useful to detect a *Brute Force* attack when trained on the *SQL injection* and is also able to detect the *XSS* when using the *Brute Force* attacks. The scores here were higher on the diagonal for the CIC-IDS-2018 than the CIC-IDS-2017.

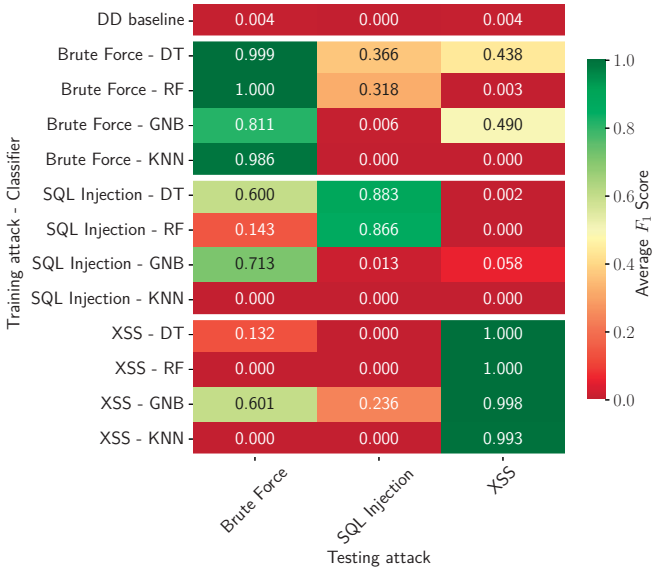


Figure 4.3: Classifier performance in detecting known and novel web attacks on the CIC-IDS-2018 dataset

Figure 4.4 shows the average F_1 scores achieved by the classifiers when detecting novel and known (D)DoS attacks. The diagonal of this matrix shows again the F_1 scores of the closed-world assumption, also obtainable from Table 4.6, while the off-diagonal values were the scores of detecting novel attacks. In this open-world setting, we observe that *Botnet* attacks were hard to detect, and neither can they easily be used to detect other variants. However, there were situations where classifiers could detect novel variants. This is, however, not symmetrical: learning attack A and finding attack B does not mean it works the other way around.

Let us now look at the results of the CIC-IDS-2018 dataset containing (D)DoS attacks. Figure 4.5 shows the results of the same experimental setup performed on the CIC-IDS-2018. Similar results were observable on the diagonal: ML algorithms could detect attacks it has trained on. In these results, it is less apparent that learning one (D)DoS attack leads to the model being able to detect another attack. Only a few combinations of train and test attacks were successful. For example, learning the *High Orbit Ion Cannon* (HOIC) with the KNN model results in high scores for testing on the *LOIC* and the *Hulk*. Results showed that classifiers such as DT and RF could not learn sufficiently from the training data, as a striking class imbalance between benign and the attack led to low performance. Still, the same observation as in the CIC-IDS-2017 is apparent: when training on attack A and being able to detect B, it does not imply the reverse also holds.

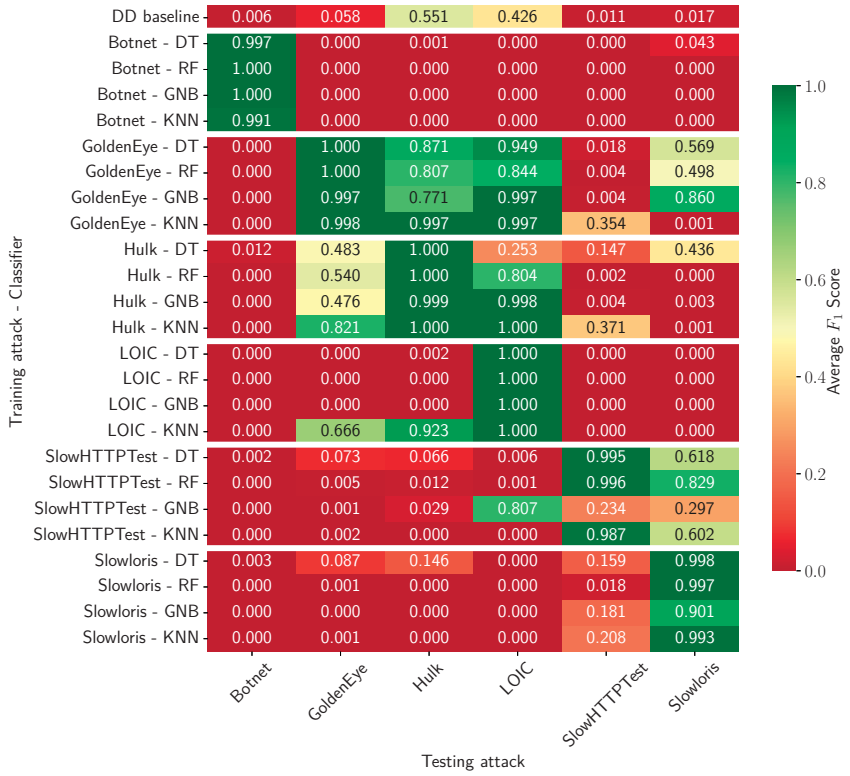


Figure 4.4: Classifier performance in detecting known and novel (D)DoS attacks on the CIC-IDS-2017 dataset

Despite the 2017 and 2018 datasets having the same cyberattacks, the HTTP interactions of the attacks are not identical. Figure 4.6 shows the results of the selected ML classifiers trained to detect a web cybervariant of the CIC-IDS-2017 and tested whether the classifiers could detect CIC-IDS-2018 web variants. We observe that there is no clear consistency between the ML models and whether they can detect known or novel attacks. Again, we see there are no symmetric results observable in the heatmap. It is not conclusive that the web attacks in the CIC-IDS-2017 are identical to the CIC-IDS-2018.

Figure 4.7 shows the results of the selected ML classifiers trained to detect a (D)DoS variant of the CIC-IDS-2017 and tested whether the classifiers could detect CIC-IDS-2018 (D)DoS variants. It can be observed that in almost no situations the classifiers could do so. The *Slowloris* attack is an exception, which performs well on all models except the GNB. This indicates that HTTP interactions were not identical despite the datasets containing the same attacks.

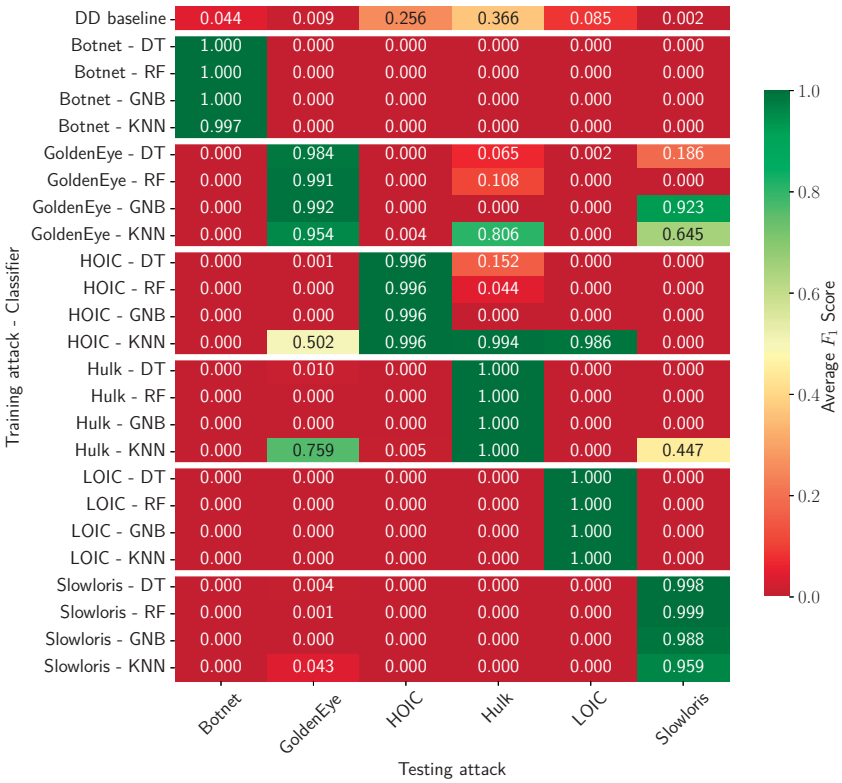


Figure 4.5: Classifier performance in detecting known and novel (D)DoS attacks on the CIC-IDS-2018 dataset

4.5.3 Learning on a Set of Variants to Detect a Novel Variant

In our last experiment, we studied which combination of cyberattacks in the learning phase results in the highest novel detection rate. Table 4.7 shows the results when classifiers were trained on one or more web variants to detect a novel web variant. The highest score achieved across the models is indicated in bold and in the last column, the set of attacks that resulted in the bold score is stated. We observe that, for both datasets, the KNN is practically useless for detecting novel web variants. The *Brute Force* attack is always used in the training dataset to achieve the highest novel detection rate.

The same procedure and analysis were performed for the (D)DoS variant. Table 4.8 shows the results of the classifiers using a set of attacks to learn from and the corresponding combination of attacks that led to the highest performance. The highest score across the models is highlighted in bold. Even though models can use more attacks to detect a novel variant, it is not necessarily the case that this

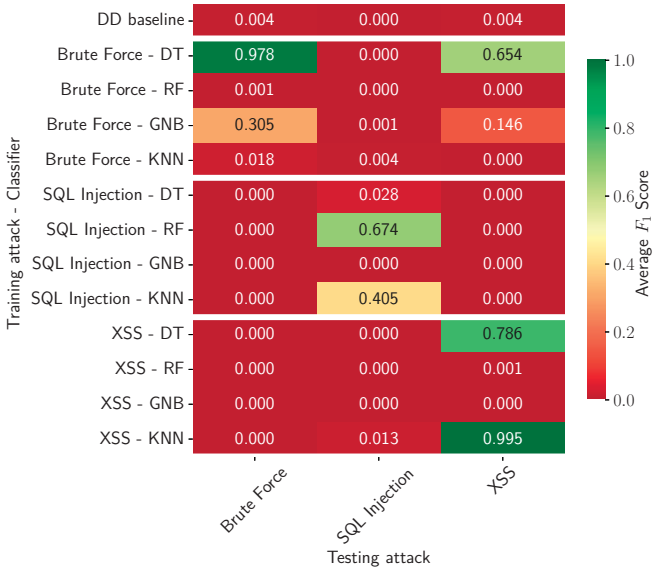


Figure 4.6: Classifier performance trained on CIC-IDS-2017 web attacks to detect CIC-IDS-2018 web attacks

Table 4.7: Maximum F_1 score for each classifier trained on multiple malicious classes to detect a novel web attack

Dataset	Attack	DD baseline	DT	GNB	KNN	RF	Train Set Opt Model
CIC-IDS-2017	Brute Force	0.054	0.202	**0.967**	0.000	0.100	{XSS}
	SQL Injection	0.000	0.257	**0.400**	0.000	0.000	{Brute Force, XSS}
	XSS	0.014	0.144	0.140	0.000	**0.327**	{Brute Force}
CIC-IDS-2018	Brute Force	0.004	0.600	**0.767**	0.000	0.152	{SQL Injection, XSS}
	SQL Injection	, 0.000	**0.366**	0.236	0.000	0.318	{Brute Force}
	XSS	0.004	0.438	**0.735**	0.000	0.200	{Brute Force, SQL}

yields the highest detection rate: even a few cyberattack classes were enough to obtain the highest performance. For the CIC-IDS-2017 the KNN model is dominantly getting the highest average F_1 scores, while for the CIC-IDS-2018 it is the GNB model. In neither case does the RF model outperform other models, which is unexpected as this model outperforms other models in detecting known attacks. For the CIC-IDS-2017 dataset, the *Hulk* attack is almost always used to obtain the highest scores with the least number of attacks required. The strong imbalance affects the DT and the RF learning process, similar to the results in experiment 2. These models could have been improved by downsampling benign entries to balance the training classes.

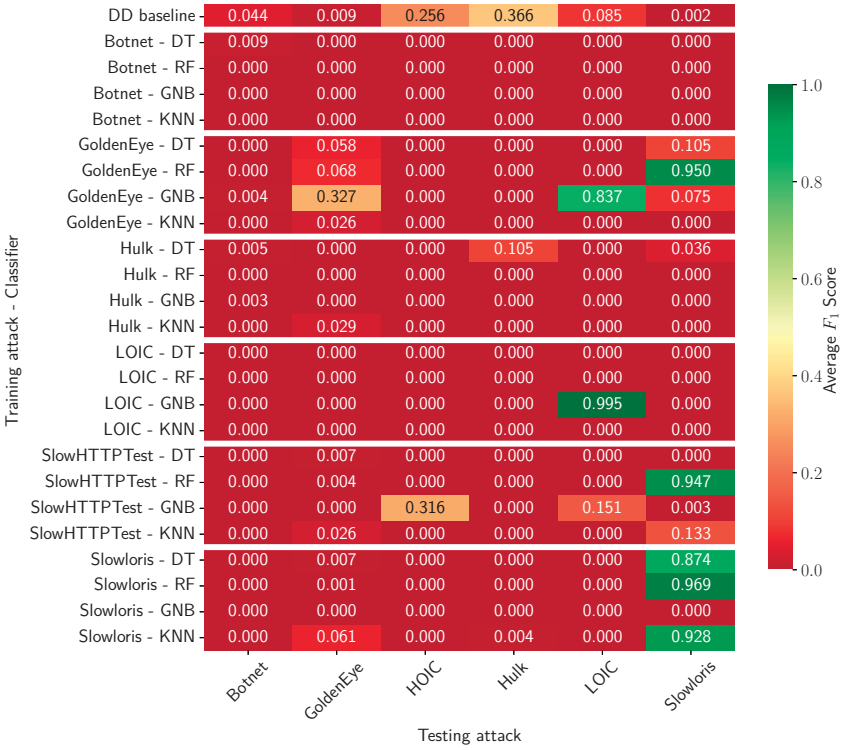


Figure 4.7: Classifier performance trained on CIC-IDS-2017 (D)DoS attacks to detect CIC-IDS-2018 (D)DoS attacks

Table 4.8: Maximum F_1 score for each classifier trained on multiple malicious classes to detect a novel (D)DoS attack

Dataset	Attack	DD baseline	DT	GNB	KNN	RF	Train Set Opt Model
CIC-IDS-2017	Botnet	0.006	**0.460**	0.291	0.000	0.000	{Hulk, LOIC, Slowloris}
	GoldenEye	0.058	0.664	0.476	**0.821**	0.782	{Hulk}
	Hulk	0.551	0.870	0.986	**0.997**	0.833	{GoldenEye, LOIC}
	LOIC	0.426	0.949	0.998	**0.999**	0.999	{Hulk}
	SlowHTTPTest	0.011	0.240	0.181	**0.399**	0.100	{Hulk, Slowloris}
	Slowloris	0.017	**0.878**	0.860	0.601	0.874	{Bot, Eye, Hulk, HTTP}
CIC-IDS-2018	Botnet	0.044	0.000	0.000	0.000	0.000	-
	GoldenEye	0.009	0.290	**0.862**	0.773	0.100	{LOIC, Hulk, Slowloris}
	HOIC	0.256	0.000	**0.853**	0.500	0.000	{LOIC, Hulk}
	Hulk	0.366	0.899	**0.999**	0.997	0.986	{GoldenEye, Slowloris}
	LOIC	0.085	0.100	0.288	**0.985**	0.000	{HOIC}
	Slowloris	0.002	0.539	**0.922**	0.837	0.000	{GoldenEye}

4.6 Discussion and Conclusion

This research provides a procedure to construct ID datasets combining multiple layers with Zeek. Zeek generates a set of extensive log files and two of them are selected to create an ML-admissible dataset for detecting cyberattacks. This procedure to create such a dataset is not limited to only these protocols but can be extended to also combine other protocols, such as TCP connection with *file transfer protocol* (FTP) interactions.

This research aimed to test to what extent ML classifiers can detect novel variants of known intrusions. A set of classifiers was applied in three different experimental setups, and we studied their ability to detect variants. The focus of this research was to study the detection of variants of (D)DoS and web attacks, but the same analysis can be performed on variants of another cyberattack. It has been shown in the first experiment that ML classifiers are to a great extent able to detect known (D)DoS attacks in a closed-world setting. For web attacks, the classifiers could not distinguish benign from malicious variants in all situations. Especially detecting *SQL Injection* instances with a GNB or KNN model was unsuccessful.

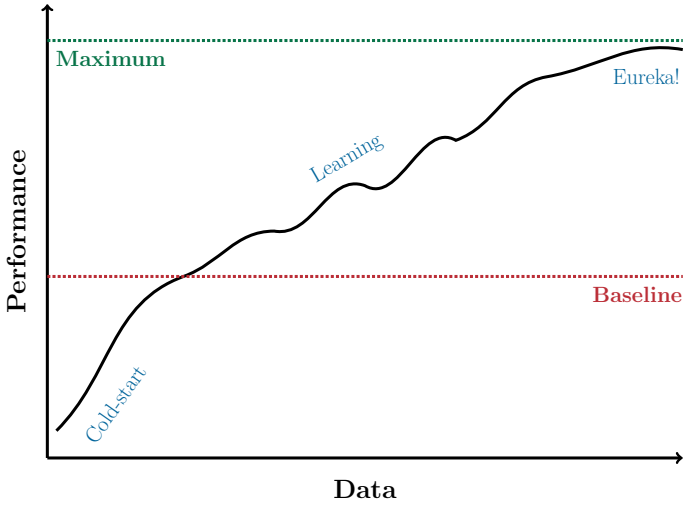
In the second experiment, it was observed that there are scenarios in which classifiers can detect a novel variant when trained on a different variant. Detecting novel variants is, however, not a two-way street: learning to detect attack A and being able to also detect attack B does not have the property that it is symmetrical. We observed that for the CIC-IDS-2017 dataset, the classifiers had a higher novel detection rate for (D)DoS variants than the results achieved on the CIC-IDS-2018. This was remarkable as the CIC-IDS-2018 contained similar attacks and more instances. However, it has been shown that the attacks are not identical between the two datasets. Only the Slowloris attack had comparable results.

The third experiment showed that it is not necessary to use many malicious variants to detect a novel attack. Sometimes, a few known attacks can lead to the highest detection rate. Looking at the results of the (D)DoS attacks, DT and RF perform poorly in detecting novel attacks. The high imbalance in the training data caused this effect. The GNB model seemed more robust against this high imbalance in the training dataset and still achieved reasonable detection rates. The results varied much for web attacks between the model and cyberattack variant combination. The KNN model turned out to be effective in detecting known *Brute Force* and *XSS* attacks but was useless to detect novel web attacks.

To conclude, this research shows that ML models can detect cyberattacks nearly as effectively as signature-based methods when given sufficient training data, and can also identify novel variants. Selecting the right ML model and a targeted set of intrusion classes for training can improve the novel intrusion detection rate.

Part III

Learning and Breakthroughs



5 The Dutch Scaler Performance Indicator: How Much Did My Model Actually Learn?

Contents

Based on [27]:
E.P. van de Bijl, J.G. Klein, J. Pries, S. Bhulai, and R.D. van der Mei, "The Dutch Scaler performance indicator: How much did my model actually learn?", *Journal of Classification*, early access, pages 1-21, 2025.

Abstract

Evaluation metrics provide a means for quantifying and comparing performances of supervised learning models, but drawing meaningful conclusions from acquired scores requires a contextual framework. In this chapter, we address this by introducing the *Dutch Scaler*, a novel performance indicator for binary classification models. It quantifies a model's learning by contextualizing empirical metric scores with a baseline (*Dutch Draw*) and a new instrument (*Dutch Oracle*) representing the prediction quality of an 'optimal' classifier. The *Dutch Scaler performance indicator* expresses the relative contribution of these components to obtain a model's score, specifying the actual learning quality. We derived closed-form expressions to map metric scores to Dutch Scaler scores for common evaluation metrics and categorized them by their functional form and second derivative. The Dutch Scaler enhances the assessment of classifiers and facilitates a framework to compare prediction quality differences between models with varying metric scores.

5.1 Introduction

"*How much did my model actually learn?*" is a fundamental question that should be answered during the development of any statistical or machine-learning model. But how do we define *learning* in this context? A formal definition is given in Mitchell [3] who states that a machine *learns* by utilizing experience (data) to improve its performance on a certain task. If we look at classification problems, the task of a model is to learn to map input data to labels. More precisely, the goal is to approximate the mapping function so well that the classifier makes accurate predictions for newly acquired input data. Checking whether this goal is met before a classifier is deployed is crucial, but how can one do so?

The performance of a classifier and the quality of its predictions are commonly expressed in *performance metrics* scores, such as the F_β *score* and *accuracy* [12], [13]. For decades, various domains have proposed evaluation metrics for classification problems highlighting one or more aspects of the confusion matrix and expressing a model's performance in a single value. Earlier research endeavors have been dedicated to studying and comparing performance metrics. For instance, Sokolova and Lapalme [66] analyzed the invariance of performance metrics by manipulating individual and total counts within the confusion matrix. Several studies provide overviews of created metrics and identify relationships, properties, and dependencies between them [84], [85]. Others compared metrics by analyzing their characteristics analytically or experimentally [54], [86]. Meanwhile, Brzezinski *et al.* [87] present a comprehensive list of ten desirable properties for performance metrics, though under varying class distribution scenarios. In more recent work, Gösgens *et al.* [88] identify three essential properties that performance metrics can potentially exhibit: monotonicity, distance, and a constant baseline. Unfortunately, no single metric simultaneously fulfills all three of these properties.

Some argue specific metrics should be preferred over others when evaluating a binary classifier on a certain problem [89]–[91]. However, no metric/measure is objectively 'the best' for all situations, so multiple metrics, and thus aspects, should be considered when assessing the quality of a classifier. Furthermore, obtained performance metric scores should be averaged over multiple (test) datasets or random seeds, as relying on a single instance could be misleading. After all, even a broken clock can be right twice a day. Moreover, considering these scores alone is insufficient to draw meaningful conclusions without a frame of reference. To illustrate this, consider the following example. Suppose a classifier obtains an average F_1 score of 0.9 and its corresponding baseline is 0.89. How would we quantify how much the model has learned when evaluated on this selected performance metric? In addition, what if, due to data quality issues such as data

noise and/or the stochastic nature of a selected classifier, an expected F_1 score of the highest metric score of 1.0 on a test dataset is unattainable?

Performance indicators are higher-order performance metrics incorporating the notion of performance bounds in quantifying a model's prediction quality. They facilitate the further comprehension and comparison of performance metrics [92]. Conceptually, it is a metric that provides insight into a goal or a predetermined baseline [93]. Based on the available literature, only one study proposes an indicator called the *accuracy barrier* [ACCBAR 45]. This indicator assesses whether the performance of a classifier is close to that of a dummy classifier (e.g., labeling only positive or negative). The ACCBAR contextualizes the performance metric *accuracy* by subtracting a baseline (the *null error rate*, NER, or the *no information rate*, NIR) from obtained evaluation scores. This approach, however, has limitations as it focuses merely on the *relative performance* compared to the baseline, but it does not quantify the model's absolute prediction quality. Moreover, it is worth considering why this indicator function does not incorporate the upper bound. In conclusion, while the ACCBAR can contextualize metric scores, it raises questions about the need to explore performance indicators further to provide a more sophisticated assessment of classification model performance. There is still room for improvement in how we measure and interpret model performance.

This chapter proposes a novel performance indicator for binary classification problems called the *Dutch Scaler* (DS). The DS quantifies the prediction quality of a classifier through a derived metric referred to as the *Dutch Scaler performance indicator* (DSPI). To contextualize metric scores, the DS employs two key components: the *Dutch Draw* (DD) baseline, which is an input-independent baseline approach indicating the expected metric score of a stochastic model that does not learn from data (as discussed in Chapters 2 and 3), and the *Dutch Oracle* (DO). This method represents and marks the prediction quality of an 'optimal' classifier in the context of selected performance metrics. These components serve as reference points for determining the DSPI.

The DS allows us to perform comparative analyses of classifiers on performance metrics as we harmonize evaluation scores of diverse metrics into a uniform reference framework. The DS is: (1) applicable for many commonly used performance metrics in binary classification problems, (2) reproducible and simple, (3) contextualizing using relevant performance thresholds, and (4) crucial to better assess performance metric scores. The DS is a valuable addition to the existing body of literature, providing an indispensable framework for improving the interpretation of realized performance metric scores. In conclusion, the DS is an essential instrument for the data science toolbox.

Our contributions are as follows: (1) we introduce the DS and show how it quantifies how much a model learned by integrating the DD baseline and the upper-performance value as performance bounds, (2) we provide closed-form expressions, mathematical properties, and parameters specifications of the DS for a set of performance metrics to convert metric scores into DSPI scores and summarize them in several tables, (3) we visually show the functions mapping metric scores to DSPI scores and categorize them on their concavity, (4) we demonstrate how the DS can be used in practice to compare classifiers and contrast the DS with the ACCBAR, and (5) we made the DS available in a Python package [32].

The organization of this chapter is as follows. Section 5.2 provides preliminaries for binary classification and briefly discusses the DD baseline. Sections 5.3 and 5.4 introduce the DO and DS, respectively. Section 5.5 gives a concavity analysis on the DS metric score transformations. Section 5.6 provides a comparative study between DSPI scores and ACCBAR scores and shows how the DSPI can be used to compare classifiers when selecting multiple metrics. Section 5.7 concludes with a discussion and future research directions.

5.2 Preliminaries

In this section, we provide the mathematical notations for binary classification problems, discuss how prediction quality is quantified using evaluation metrics, and define how one can achieve performance optimality. We then elaborate on the DD baseline and provide some necessary notations for constructing it.

5.2.1 Binary Classification

The task in a *binary classification problem* is to let a classifier learn the relationship between input data and a binary output vector. We consider a set of $M \in \mathbb{N}^+$ instances of the form $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_M, y_M)\}$, where, without loss of generality, $\mathbf{x}_i \in \mathbb{R}^d$ is a d dimensional feature space and $y_i \in \{0, 1\}$ is its corresponding label. Observations with response value 1 are 'positive', while those with 0 are 'negative'. We denote the whole dataset as $(\mathbf{X}, \mathbf{y})$ with $\mathbf{X} \coloneqq (\mathbf{x}_1 \ldots \mathbf{x}_M)^\top \in \mathbb{R}^{M \times d}$ and $\mathbf{y} \coloneqq (y_1, y_2, \ldots, y_M)$. Let P denote the number of positive instances and N denote the number of negative instances. By definition, $P + N = M$.

5.2.2 Performance Quantification

By comparing the labels of a set of predicted instances, denoted by $\hat{\mathbf{y}}$, with their actual labels, four *base measures* can be constructed: the number of *true positives*

(TP), *true negatives* (TN), *false negatives* (FN), and *false positives* (FP). Let us denote $z : \mathbf{y} \times \hat{\mathbf{y}} \mapsto \mathbb{N}^4$ as the function to derive those four base measures. We define $\hat{P} = \text{TP} + \text{FP}$ as the number of predicted positive instances and $\hat{N} = \text{TN} + \text{FN}$ as the number of the predicted negatives.

Performance metrics are derived from the before-mentioned base measures. Let us define a performance metric as $\mu : \mathbb{N}^4 \mapsto \mathbb{R}$, mapping a function of the four base measures to a performance metric score. A metric does not necessarily require all base measures as input values, so only those used are included in the functional notation throughout this chapter. Let us denote a composition function deriving the performance metric score directly from the predicted and the actual label vectors as $w = \mu \circ z : \mathbf{y} \times \hat{\mathbf{y}} \mapsto \mathbb{R}$. In this chapter, we consider the same performance metrics as stated in Chapter 2, and their definitions and co-domains can be found there.

5.2.3 Performance Optimality

The task in a classification problem is to find the 'optimal' classifier, which makes accurate predictions. In Chapter 3, three important factors are stated to specify 'optimality': (1) as binary classifiers can be stochastic, we typically examine their *expected* performance; (2) the 'optimal' classifier is the best in the set of considered classifiers; and (3) this classifier should be the 'best' for a specific dataset, but all permutations of this dataset should be considered to prevent a coincidental perfectly prediction by a deterministic classifier. 'Best' and 'optimal' here depend on the desire to maximize or minimize one or more selected performance metrics. For example, *accuracy* is a metric we would naturally like to maximize, while the *false discovery rate* is preferably minimized. Assuming that the selected metrics are naturally maximized, we provide derivations only for maximization. For naturally minimized metrics, one can follow the procedure below by substituting 'maximizing' with 'minimizing' or simply multiplying the metric with '-1'.

A *binary classifier* is defined as a function $h : \chi \times \mathbb{R} \to \{0, 1\}$ that maps feature values to zero or one, where the second input is used to capture the randomness nature in a stochastic model, often referred to as the *random seed*, see Chapter 3. This classifier can only label one instance at a time, so to classify multiple instances simultaneously, we define $h_M : \chi^M \times \mathbb{R} \to \{0, 1\}^M$ as the function that predicts $M \in \mathbb{N}^+$ instances. Any single instance classifier h can be extended to predict M instances simultaneously by applying this classifier for each instance individually. Let us define H_M as the set of all possible classifiers of the form h_M.

The objective in searching for the 'best' classifier is to identify the so-called *average-permutation-optimal* classifier for a selected performance metric. This

means we try to find the classifier that optimizes the expected performance score, accounting for the variations due to data permutations and classifier randomness. Let us denote π as a function that permutes data in all possible permutations, as specified in Chapter 3. If we selected a performance metric μ, the *average-permutation* expectation for a single classifier h_M can be derived as follows:

$$\mu_{h_M} \coloneqq \mathbb{E}_{\pi}[\mathbb{E}_{r \in \mathbb{R}}[w(h_M(\mathbf{X}_{\pi}, r), \mathbf{y}_{\pi})]],$$

where w is the composition function of μ, as described in Section 5.2.2. To find the *average-permutation-optimal* classifier, we have to search for the classifier with the highest *average-permutation* expectation in the set of all possible classifiers H_M, or mathematically $h_M^{\max} \in \arg\max_{h_M \in H_M} \mu_{h_M}$. We define the *average-permutation-optimal expectation score* for performance metric μ as follows:

$$\Omega_{\mu} \coloneqq \mathbb{E}_{\pi}[\mathbb{E}_{r \in \mathbb{R}}[w(h_M^{\max}(\mathbf{X}_{\pi}, r), \mathbf{y}_{\pi})]].$$

The *average-permutation* expectation cannot always be determined due to the generalization error, but we can look at the empirical estimator: $\bar{\mu}$. Suppose we have a random classifier h'_M, and we apply it K times on a dataset with different seeds. The empirical estimator can be determined by

$$\bar{\mu} \coloneqq \frac{\sum_{r=1}^{K} w(h'_M(\mathbf{X}, r), \mathbf{y})}{K}.$$

This empirical value is the score we want to contextualize with the performance bounds in our performance indicator. The difference between Ω_{μ} and $\overline{\mu}$ is also known as the *generalization error*. The generalization error is, unfortunately, inevitable. The *probably approximately correct* (PAC) learning framework offers theorems that estimate the number of instances needed to determine the likelihood that the error remains within a specified bound. While this framework can guide the selection of K, there is no definitive rule of thumb; generally, a higher K tends to yield better results. The following sections discuss the two critical components for deriving essential performance bounds: a baseline and the performance score of the 'optimal' classifier.

5.2.4 Dutch Draw Baseline

The selected method for deriving a binary classification baseline for a given metric is the DD, as discussed in Chapter 2 and 3. We selected this method as it is the best binary classifier that does not learn from data (input-independent), as proven in Chapter 3. The DD baseline is established by deriving the 'optimal' DD classifier, which optimizes the expected value of the selected performance metric. A DD classifier predicts M instances by randomly assigning the value '1'

to $d = \lfloor M \cdot \theta \rceil$ instances and the value '0' to the remaining $M - d$ instances. Here, $\lfloor x \rceil$ denotes rounding x to the nearest integer (e.g., 1.4 rounds to 1 and 2.6 rounds to 3). The parameter $\theta \in [0, 1]$ specifies the proportion of M predicted as positive. Since this random classifier depends solely on θ, it is considered data-independent and does not learn.

Let us provide a mathematical formulation for the DD classifier. Let $S \coloneqq \{0, 1\}^M$ denote the set of all possible binary vectors to predict M instances simultaneously. We can decompose this set S into disjoint sets such that all vectors contain the same number of ones in each set. Let us thus define $S_k \coloneqq \{s \in S | \sum_{i=1}^{M} s_i = k\}$ with $k \in \{0, 1, \ldots, M\}$. It holds that $\cup_{k=0}^{M} S_k = S$. A DD classifier, denoted by $\sigma_\theta : \mathbb{N} \mapsto \{0, 1\}^M$, randomly draws one vector from one of these decomposed sets (S_k) and is mathematically defined as $\sigma_\theta(M) \sim \text{Uniform}(S_{\lfloor M \cdot \theta \rceil})$ where $\text{Uniform}(A)$ is defined as the uniform distribution over a set A and $\lfloor M \cdot \theta \rceil$ specifies the number of positively predicted instances $(\hat{P})$ for this classifier. The distributions of the base measures, denoted by $\text{TN}_\theta^{\text{DD}}$, $\text{TP}_\theta^{\text{DD}}$, $\text{FN}_\theta^{\text{DD}}$, and $\text{FP}_\theta^{\text{DD}}$, are directly determined by σ_θ and all follow a hypergeometric distribution with parameters depending on M, P, and $\lfloor M \cdot \theta \rceil$.

It can be observed in this mathematical notation that multiple values of θ can result in the same classifier. For example, suppose $M = 10$ and $\theta_1 = 0.1$ and $\theta_2 = 0.11$, then $\sigma_{\theta_1}(M) = \sigma_{\theta_2}(M)$. Let us, therefore, introduce another variable $\theta^* \coloneqq \frac{\lfloor M \cdot \theta \rceil}{M}$ as the discretized version of θ. This θ^* reduces the search space for finding the set of θ, leading to the optimal DD. Furthermore, we define

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

as the set of all unique values that θ^* can obtain for all $\theta \in [0, 1]$. There are, however, some limitations to $\hat{P}$ or $\hat{N}$. For example, we require $\hat{P} > 0$ to have a defined precision score. Let us, therefore, introduce Θ_μ^* as the set of possible θ^* values respecting the limitations of the evaluation metric μ. The DD baseline is the expectation of the optimal DD classifier for a selected performance metric μ. Let us define θ_{opt}^* as the θ^* leading to this optimum. In mathematical terms, we try to find

$$\theta_{\text{opt}}^* \in \Theta_{\text{opt}}^* \coloneqq \underset{\theta^* \in \Theta_\mu^*}{\arg\max} \left\{ \mathbb{E}\left[w(\sigma_{\theta^*}(M), \mathbf{y}) \right] \right\}.$$

The DD baselines and corresponding θ_{opt}^* values for all selected evaluation metrics μ can be found in Chapter 2.

5.3 Dutch Oracle

Deriving an exact upper bound on the expected performance of supervised learning models is often challenging for several reasons. First, its performance is bounded by the quality and size of the used data [94]. The existence of noise in labels or input data, class imbalance, sample bias, and outliers illustrate why classifiers cannot always make perfect predictions [95]. Secondly, there does not exist one unique model that achieves the highest possible performance score for all problems as described in the no-free-lunch theorem [96]. Therefore, we would have to search the set of all possible classifiers to find the model achieving the 'best' score for a specific problem. Thirdly, there are an infinite number of possible classifiers. Still, we can only gather empirical results for a finite number of classifiers, so it is always possible that we did not consider a model that approximates the target function better. Upper bounds should represent expected performances, while with empirical results, we can only estimate the expectations.

Fortunately, we can approximate the performance of an 'optimal' classifier using an abstraction from the *active learning* domain called *oracle* [97]. In general, an oracle is an entity that knows the correct answer to all questions [98]. In a classification problem, this would be a model that always predicts the correct label. But, as mentioned in Raykar *et al.* [98], even an oracle sometimes makes mistakes or is not always correct, introducing the notion of an *imperfect oracle*. This imperfect oracle can approximate the expected performance score of the optimal model by finding the right balance of correct and incorrect predictions.

We propose a novel (im)perfect oracle classifier called the DO. The DO can be seen as a proxy for optimal model performance, establishing the theoretical performance limit for any chosen metric. The imperfect oracle enables the incorporation of upper limits when determining prediction quality scores, providing more nuanced evaluations of a classifier's performance. The DO approximates optimal model performance by balancing correct predictions with occasional errors. The DO makes an incorrect prediction for each instance with a probability $\rho \in [0, 1]$. As such, each base measure can be represented as the outcome of a binomial trial. Thus, we get

$$\mathrm{TP}_\rho^{\mathrm{DO}} \sim B(P, (1-\rho)), \quad \mathrm{FN}_\rho^{\mathrm{DO}} \sim B(P, \rho),$$

and

$$\mathrm{TN}_\rho^{\mathrm{DO}} \sim B(N, (1-\rho)), \quad \mathrm{FP}_\rho^{\mathrm{DO}} \sim B(N, \rho).$$

Independently of the actual label of the instance, the DO will make a mistake with this probability ρ. Deriving expectations of these base measures with the DO gives us the following:

$$\mathbb{E}\left[\mathrm{TP}_\rho^{\mathrm{DO}}\right] = P \cdot (1-\rho), \quad \mathbb{E}\left[\mathrm{FN}_\rho^{\mathrm{DO}}\right] = P \cdot \rho,$$

and

$$\mathbb{E}\left[\mathrm{TN}_\rho^{\mathrm{DO}}\right] = N \cdot (1-\rho), \quad \mathbb{E}\left[\mathrm{FP}_\rho^{\mathrm{DO}}\right] = N \cdot \rho.$$

Exploring the extremes of the prediction quality generated by the DO proves to be insightful. Choosing $\rho = 0$ results in a flawless DO, which we call the *perfect predictor* (PP). The PP, a classical approach, accurately predicts the label for a given instance, making it suitable for approximating the maximum performance score when a model consistently achieves perfect predictions. Conversely, opting for $\rho = 1$ leads to a DO prone to mere errors, labeled the *terrible predictor*. Instances presented to this oracle consistently receive incorrect labels. However, since the objective is to estimate the score of the optimal model, this oracle lacks practical value, with any baseline model expected to outperform it.

Let us outline two ways to determine ρ. First, it can be determined through one expert straightforward estimation, providing an approximate but reasonably accurate assessment of the upper limit. Although not as precise as rigorous quantitative analysis, the user-driven estimation method is quick and practical, making it a valuable tool for setting the upper boundary, especially when more rigorous data or analysis is not readily available.

Second, a more quantitative approach involves using theoretically derived upper bounds on classifier performance. One such approach is determining the Bayes error rate [99], [100], representing the minimum error rate that any classifier can theoretically achieve. Once the Bayes error is known, the corresponding expected metric score for this 'optimal' model can be calculated. For instance, if the Bayes error rate for *accuracy* is 0.15, then the expected accuracy of the 'optimal' classifier would be 0.85. The corresponding ρ to align the DO under the DS is 0.15, but in Section 5.4.6 and specifically Table 5.4, we will give exact formulas to translate other theoretical 'optimal' model scores to ρ. Still, depending on the problem instance and the theoretical approach chosen, other bounds—such as those derived from PAC learning theory—can provide quantitative values to determine the upper limit of an 'optimal' classifier's performance.

5.4 Dutch Scaler

This section describes the DS and how it quantifies how much a classifier has learned. First, we discuss the essential characteristics an indicator should possess to achieve this purpose. Second, we mathematically define the DS and show under which conditions our indicator satisfies these desirable properties.

5.4.1 Desirable Properties of an Indicator

Before we discuss our proposed performance indicator, let us outline the two key characteristics it should possess.

Normalization and Range $[0, 1]$: If a classifier's metric score surpasses a baseline of a selected performance metric and does not exceed the expected score of the 'optimal' model, then the indicator score should fall within the range $[0, 1]$. In this context, '0' signifies the baseline performance, while '1' represents the upper limit of the expected performance. These bounds should be based on expectations rather than empirical averages, as the latter lacks theoretical grounding and may be harder to interpret or extend to new situations. On the one hand, if a classifier's performance falls short of the baseline, the indicator score should be negative. This negative score signals that the classifier underperforms in comparison to the baseline. Identical to a negative R^2 in a regression context, its numerical value lacks practical significance because it indicates that the regressor's performance is inferior to predicting the mean. On the other hand, if the empirical metric score is above the expectation of an 'optimal' model, indicating a generalization error, the derived score will be above 1. An interpretation of this indicator score can be either (1) the expectation of the 'optimal' model is incorrect, or (2) the empirical estimation of the expected metric score is deviant from the actual expectation of the corresponding model. The overly-optimistic empirical acquired score introduces the generalization error in the latter situation. In either scenario, the values should be carefully re-evaluated.

Strictly Increasing with Performance: Indicator scores must exhibit a strict positive monotone relationship with the realized scores. As a metric score increases, its corresponding indicator score should also consistently rise. Failing to demonstrate this relationship would be counterproductive, as it would mean that striving for better performance does not yield higher indicator scores.

These properties contribute to form a clearly defined, easily interpretable performance indicator that provides valuable insights for evaluating and comparing classifier performance.

5.4.2 Definition and Objective

The DS quantitatively contextualizes metrics scores by redefining the computation of the base measures to determine how much a model has learned. The DS base measures are derived using a convex combination of the expectations of the DD baseline and DO base measures. The mathematical definition of each DS base measure is expressed as follows:

$$\begin{aligned}
\mathrm{TP}_\alpha &:= \alpha \cdot \mathbb{E}\left[\mathrm{TP}_\rho^{\mathrm{DO}}\right] + (1-\alpha) \cdot \mathbb{E}\left[\mathrm{TP}_{\theta^*_{\mathrm{opt}}}^{\mathrm{DD}}\right], \\
\mathrm{TN}_\alpha &:= \alpha \cdot \mathbb{E}\left[\mathrm{TN}_\rho^{\mathrm{DO}}\right] + (1-\alpha) \cdot \mathbb{E}\left[\mathrm{TN}_{\theta^*_{\mathrm{opt}}}^{\mathrm{DD}}\right], \\
\mathrm{FN}_\alpha &:= \alpha \cdot \mathbb{E}\left[\mathrm{FN}_\rho^{\mathrm{DO}}\right] + (1-\alpha) \cdot \mathbb{E}\left[\mathrm{FN}_{\theta^*_{\mathrm{opt}}}^{\mathrm{DD}}\right], \\
\mathrm{FP}_\alpha &:= \alpha \cdot \mathbb{E}\left[\mathrm{FP}_\rho^{\mathrm{DO}}\right] + (1-\alpha) \cdot \mathbb{E}\left[\mathrm{FP}_{\theta^*_{\mathrm{opt}}}^{\mathrm{DD}}\right].
\end{aligned}$$

Here, $\alpha \in [0, 1]$ is the DSPI. The DSPI controls the weight assigned to the DO and DD base measures expectations. Metric scores (μ) are derived from one or more of the four base measures, as discussed in Section 5.2.2. Similarly, metrics under the DS can be computed using the same transformation, and we denote its computation as $\mu_\alpha := \mu(\mathrm{TP}_\alpha, \mathrm{TN}_\alpha, \mathrm{FP}_\alpha, \mathrm{FN}_\alpha)$.

The objective of the DS is to find the DSPI score (α) that results in the corresponding acquired metric score. Mathematically, find α such that $\mu_\alpha = \overline{\mu}$, where $\overline{\mu}$ is a realized performance metric score described in Section 5.2.3. The intuition behind this approach is that we are interested in how much contribution of the DO and the DD baseline is required to achieve an identical empirical metric score. The underlying question is: *"How much learning is needed to achieve the same score?"*. There are metrics, such as the *true positive rate*, where the DD baseline directly results in the 'optimal' model's expected metric score. The highest possible score can be achieved for these metrics without learning from the data. We focus on metrics where the DD baseline does not result in the 'optimal' model score and where a model's performance lies between the DD baseline and the DO. Mathematical reverse engineering is needed to derive the required α, which we will show in the following sections.

5.4.3 Substitutions and Properties

Having established the definitions of the base measures under the DS, we can now formulate the performance metrics using these base measures. Let us substitute the expectations of the DD baseline and the DO in the definition of the DS as stated in Sections 5.2.4 and 5.3. From Chapter 2, it follows that $\mathbb{E}\left[\mathrm{TP}_{\theta^*_{\mathrm{opt}}}^{\mathrm{DD}}\right] = P \cdot \theta^*_{\mathrm{opt}}$,

$\mathbb{E}\left[\mathrm{TN}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}\right] = N \cdot (1-\theta^*_{\mathrm{opt}})$, $\mathbb{E}\left[\mathrm{FN}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}\right] = P - \mathbb{E}\left[\mathrm{TP}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}\right]$ and $\mathbb{E}\left[\mathrm{FP}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}\right] = N - \mathbb{E}\left[\mathrm{TN}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}\right]$. By substituting them into these expectations in the definition of the DS base measures, we get the following:

$$\begin{aligned}
\mathrm{TP}_\alpha &:= \alpha \cdot \mathbb{E}\left[\mathrm{TP}^{\mathrm{DO}}_\rho\right] + (1-\alpha) \cdot \mathbb{E}\left[\mathrm{TP}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}\right] \\
&= \alpha \cdot P \cdot (1-\rho-\theta^*_{\mathrm{opt}}) + P \cdot \theta^*_{\mathrm{opt}}, && (5.1)\\
\mathrm{TN}_\alpha &:= \alpha \cdot \mathbb{E}\left[\mathrm{TN}^{\mathrm{DO}}_\rho\right] + (1-\alpha) \cdot \mathbb{E}\left[\mathrm{TN}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}\right] \\
&= \alpha \cdot N \cdot (\theta^*_{\mathrm{opt}}-\rho) + N \cdot (1-\theta^*_{\mathrm{opt}}), && (5.2)\\
\mathrm{FN}_\alpha &:= \alpha \cdot \mathbb{E}\left[\mathrm{FN}^{\mathrm{DO}}_\rho\right] + (1-\alpha) \cdot \mathbb{E}\left[\mathrm{FN}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}\right] \\
&= \alpha \cdot P \cdot (\rho-1+\theta^*_{\mathrm{opt}}) + P \cdot (1-\theta^*_{\mathrm{opt}}), && (5.3)\\
\mathrm{FP}_\alpha &:= \alpha \cdot \mathbb{E}\left[\mathrm{FP}^{\mathrm{DO}}_\rho\right] + (1-\alpha) \cdot \mathbb{E}\left[\mathrm{FP}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}\right] \\
&= \alpha \cdot N \cdot (\rho-\theta^*_{\mathrm{opt}}) + N \cdot \theta^*_{\mathrm{opt}}. && (5.4)
\end{aligned}$$

By definition, we have $0 \leq \mathrm{TP}_\alpha, \mathrm{FN}_\alpha \leq P$, and $0 \leq \mathrm{TN}_\alpha, \mathrm{FP}_\alpha \leq N$. As mentioned before, $\mathrm{TP}_\alpha + \mathrm{FN}_\alpha = P$ and $\mathrm{TN}_\alpha + \mathrm{FP}_\alpha = N$. A convenient mathematical expression in the derivation of closed-form expressions of performance metrics under the DS is the number of positively/negatively predicted instances, which is given by

$$\hat{P} = \mathrm{TP}_\alpha + \mathrm{FP}_\alpha = \alpha \cdot (\rho \cdot (N-P) + P - M \cdot \theta^*_{\mathrm{opt}}) + M \cdot \theta^*_{\mathrm{opt}}, \tag{5.5}$$

and

$$\hat{N} = \mathrm{TN}_\alpha + \mathrm{FN}_\alpha = \alpha \cdot (\rho \cdot (P-N) + M \cdot \theta^*_{\mathrm{opt}} - P) + M \cdot (1-\theta^*_{\mathrm{opt}}). \tag{5.6}$$

These derivations help to rewrite metric definitions in terms of the DS parameters. Table 5.1 shows the DS definition for the selected performance metrics. They are functions of α, P, M, θ^*_{opt}, and ρ. The values of M and P are inherent to the specific dataset employed, while ρ is a user-estimated parameter. We will elaborate on this later, but since multiple values of θ^*_{opt} can lead to the DD baseline, it is essential to establish criteria for selecting the appropriate θ^*_{opt} for the DS. In the following sections, we will elaborate on the selection of θ^*_{opt}, describe the mathematical reverse engineering of α, and show the restrictions on the user-estimated parameter ρ to satisfy all required properties defined for an indicator. Still, first, we will look at the range of the DS.

5.4.4 Range

The range of the metric scores (μ_α) the DS can achieve is determined by the selected metric and its specific definition. Let us formulate two theorems describing

Table 5.1: Performance metrics substituted with DS base measures

Metric	DS Abbr.	Definition under the DS
Positive predicted value	PPV_α	$\frac{\alpha \cdot P \cdot (1-\rho-\theta^*_{\text{opt}}) + P \cdot \theta^*_{\text{opt}}}{\alpha \cdot (\rho \cdot (N-P) + P - M \cdot \theta^*_{\text{opt}}) + M \cdot \theta^*_{\text{opt}}}$
Negative predicted value	NPV_α	$\frac{\alpha \cdot N \cdot (\theta^*_{\text{opt}} - \rho) + N \cdot (1-\theta^*_{\text{opt}})}{\alpha \cdot (\rho \cdot (P-N) + M \cdot \theta^*_{\text{opt}} - P) + M \cdot (1-\theta^*_{\text{opt}})}$
F_β score	F^β_α	$\frac{(1+\beta^2) \cdot (\alpha \cdot P \cdot (1-\rho-\theta^*_{\text{opt}}) + P \cdot \theta^*_{\text{opt}})}{\alpha \cdot (\rho \cdot (N-P) + P - M \cdot \theta^*_{\text{opt}}) + M \cdot \theta^*_{\text{opt}} + P \cdot \beta^2}$
Youden's J statistic/index	J_α	$\alpha \cdot (1 - 2 \cdot \rho)$
Markedness	MK_α	$\alpha \cdot (1 - 2 \cdot \rho) \cdot \frac{P \cdot N}{\hat{P} \cdot \hat{N}}$
Accuracy	Acc_α	$\frac{\alpha \cdot P \cdot (1-\rho-\theta^*_{\text{opt}}) + P \cdot \theta^*_{\text{opt}} + \alpha \cdot N \cdot (\theta^*_{\text{opt}} - \rho) + N \cdot (1-\theta^*_{\text{opt}})}{M}$
Balanced cccuracy	BAcc_α	$\alpha \cdot (1 - 2 \cdot \rho) \cdot \frac{1}{2} + \frac{1}{2}$
Matthews' correlation coefficient	MCC_α	$\alpha \cdot (1 - 2 \cdot \rho) \cdot \sqrt{\frac{P \cdot N}{\hat{P} \cdot \hat{N}}}$
Cohen's kappa	κ_α	$\alpha \cdot (1 - 2 \cdot \rho) \cdot \frac{2 \cdot P \cdot N}{\hat{P} \cdot N + \hat{N} \cdot P}$
Fowlkes-Mallow index	FM_α	$\frac{\alpha \cdot P \cdot (1-\rho-\theta^*_{\text{opt}}) + P \cdot \theta^*_{\text{opt}}}{\sqrt{P \cdot (\alpha \cdot (\rho \cdot (N-P) + P - M \cdot \theta^*_{\text{opt}}) + M \cdot \theta^*_{\text{opt}})}}$
Threat score	TS_α	$\frac{\alpha \cdot P \cdot (1-\rho-\theta^*_{\text{opt}}) + P \cdot \theta^*_{\text{opt}}}{P + \alpha \cdot N \cdot (\rho - \theta^*_{\text{opt}}) + N \cdot \theta^*_{\text{opt}}}$
G-mean 2	$\text{G}^{(2)}_\alpha$	$\sqrt{(\alpha \cdot (1 - \rho + \theta^*_{\text{opt}}) + \theta^*_{\text{opt}}) \cdot (\alpha \cdot (\theta^*_{\text{opt}} - \rho) + (1 - \theta^*_{\text{opt}}))}$

the value of μ_α at $\alpha \in \{0, 1\}$ to check whether and under which conditions the previously described desirable properties hold.

Baseline Bound: Our first desirable property of an indicator describes that an indicator score of 0 should indicate the expectation score of a baseline model. For our indicator, we selected the DD baseline as our baseline approach. The following theorem indicates the two properties metric μ_α should possess such that it holds that the DS with $\alpha = 0$ equals the DD baseline.

Theorem 2. *If μ_α is strictly increasing in $\alpha \in [0, 1]$ and μ_0 is linear in $TP^{DD}_{\theta^*_{opt}}$, then its minimum is $\mathbb{E}\left[\mu_\alpha\left(TP^{DD}_{\theta^*_{opt}}\right)\right]$, which is the DD baseline.*

Proof. It holds that μ_α is strictly increasing in α, so the minimum of μ_α is achieved for $\alpha = 0$, i.e.,

$$\min_{\alpha \in [0,1]} \mu_\alpha = \mu_{\alpha=0}(\text{TP}_{\alpha=0}, \text{TN}_{\alpha=0}) \overset{(5.1),\,(5.2)}{=} \mu_{\alpha=0}\left(\mathbb{E}\left[\text{TP}^{\text{DD}}_{\theta^*_{\text{opt}}}\right], \mathbb{E}\left[\text{TN}^{\text{DD}}_{\theta^*_{\text{opt}}}\right]\right).$$

Hence, the minimum $\mu_{\alpha=0}$ depends only on the DD. Under the DD, all four base measures can be written as linear functions of $\text{TP}^{\text{DD}}_{\theta^*_{\text{opt}}}$. We can drop the $\text{TN}^{\text{DD}}_{\theta^*_{\text{opt}}}$ argument in μ_α. Let $\mu^*_{\alpha=0}$ be this measure, i.e., $\mu^*_{\alpha=0}\left(\mathbb{E}\left[\text{TP}^{\text{DD}}_{\theta^*_{\text{opt}}}\right]\right) := \mu_{\alpha=0}\left(\mathbb{E}\left[\text{TP}^{\text{DD}}_{\theta^*_{\text{opt}}}\right], N - \lfloor M \cdot \theta^*_{\text{opt}} \rceil + \mathbb{E}\left[\text{TP}^{\text{DD}}_{\theta^*_{\text{opt}}}\right]\right)$. By the linearity of the expect-

ation operator, we have

$$\mu_{\alpha=0}\left(\mathbb{E}\left[\mathrm{TP}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}\right], \mathbb{E}\left[\mathrm{TN}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}\right]\right) = \mu^*_{\alpha=0}\left(\mathbb{E}\left[\mathrm{TP}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}\right]\right) = \mathbb{E}\left[\mu^*_{\alpha=0}\left(\mathrm{TP}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}\right)\right].$$

The right-hand side is precisely the definition of the DD baseline for the evaluation measures μ_α. □

It immediately follows from Theorem 4.1 that $\mathbb{E}\left[\mu_{\alpha=0}\left(\mathrm{TP}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}, \mathrm{TN}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}\right)\right] \leq \mu_\alpha(\mathrm{TP}_\alpha, \mathrm{TN}_\alpha)$ under the specified conditions. Some metrics, i.e., TS_α and $\mathrm{G}^{(2)}_\alpha$, do not satisfy the linearity property in $\mathrm{TP}^{\mathrm{DD}}_{\theta^*_{\mathrm{opt}}}$ under μ_0, so Theorem 4.1 does not necessarily hold for them. It is, however, possible that an equality does occur. For TS_α, inequality occurs only in cases where $P = 1$ and $\theta^*_{\mathrm{opt}} < 1$. Since no closed-form expression is currently known for determining θ^*_{opt} for $\mathrm{G}^{(2)}_\alpha$ resulting in the DD baseline, identifying when $\mu_{\alpha=0}$ equals the DD baseline remains an open question. In the case of inequality, additional considerations or adjustments are required to construct indicator scores satisfying the desirable properties of an indicator as described in Section 5.4.1.

Optimal Model Bound: Using the following theorem, let us now examine the alignment of the DS with the expected performance score of the 'optimal' classifier.

Theorem 3. *If μ_α is strictly increasing in $\alpha \in [0, 1]$, then the maximum of μ_α is equal to $\mu_{\alpha=1}(\mathbb{E}\left[\mathrm{TP}^{\mathrm{DO}}_\rho\right], \mathbb{E}\left[\mathrm{TN}^{\mathrm{DO}}_\rho\right])$.*

Proof. μ_α is strictly increasing and hence, the maximum of μ_α is achieved for $\alpha = 1$, i.e.,

$$\max_{\alpha\in[0,1]} \mu_\alpha = \mu_{\alpha=1}(\mathrm{TP}_{\alpha=1}, \mathrm{TN}_{\alpha=1}) \overset{(5.1),\,(5.2)}{=} \mu_{\alpha=1}(\mathbb{E}\left[\mathrm{TP}^{\mathrm{DO}}_\rho\right], \mathbb{E}\left[\mathrm{TN}^{\mathrm{DO}}_\rho\right]).$$

□

More specifically,

$$\max_{\alpha\in[0,1]} \mu_\alpha = \mu_{\alpha=1}(P \cdot (1-\rho), N \cdot (1-\rho)).$$

Table 5.2 gives the boundaries of a set of selected performance metrics we would naturally like to maximize. Since the considered performance metrics, except for TS_α and $\mathrm{G}^{(2)}_\alpha$, exhibit DD baseline linearity in $\mathrm{TP}^{\mathrm{DD}}_\theta$, it can be inferred that the lower bound of the DS is equivalent to the DD baseline. Notably, the alignment of

TS_α remains accurate despite its non-linearity in $\text{TP}_\theta^{\text{DD}}$. To ensure that $\mu_0 < \mu_1$, certain constraints on the parameter ρ are necessary, which we will specify in Section 5.4.6. When the realized performance score is between these bounds, we should search for the resulting indicator score, which we will discuss in the next section.

Table 5.2: Boundaries of DS performance metrics

Metric	DD baseline Θ^*_{opt}	Baseline bound (μ_0)	Optimal model bound (μ_1)
PPV_α	$\Theta^* \setminus \{0\}$	$\frac{P}{M}$	$\frac{P\cdot(1-\rho)}{\rho\cdot(N-P)+P}$
NPV_α	$\Theta^* \setminus \{1\}$	$\frac{N}{M}$	$\frac{N\cdot(1-\rho)}{\rho\cdot(P-N)+N}$
F^β_α	$\{1\}$	$\frac{(1+\beta^2)\cdot P}{M+P\cdot\beta^2}$	$\frac{(1+\beta^2)\cdot P\cdot(1-\rho)}{\rho\cdot(N-P)+P\cdot(1+\beta^2)}$
J_α	Θ^*	0	$1-2\cdot\rho$
MK_α	$\Theta^* \setminus \{0,1\}$	0	$\frac{N\cdot P\cdot(1-2\cdot\rho)}{-\rho^2\cdot(P-N)^2+\rho\cdot(P-N)^2+N\cdot P}$
Acc_α	if $P=N\rightarrow\Theta^*$ else $\rightarrow\{\lvert P<\frac{M}{2}\rvert\}$	$\frac{\max\{P,N\}}{M}$	$1-\rho$
BAcc_α	Θ^*	0.5	$1-\rho$
MCC_α	$\Theta^* \setminus \{0,1\}$	0	$\frac{\sqrt{P\cdot N}\cdot(1-2\cdot\rho)}{\sqrt{(\rho\cdot(N-P)+P)\cdot(\rho\cdot(P-N)+N)}}$
κ_α	if $P=M\rightarrow\Theta^*\setminus\{1\}$ else $\rightarrow\Theta^*$	0	$\frac{2\cdot P\cdot N\cdot(1-2\cdot\rho)}{\rho\cdot(N-P)^2+2\cdot P\cdot N}$
FM_α	$\{1\}$	$\sqrt{\frac{P}{M}}$	$\frac{\sqrt{P}\cdot(1-\rho)}{\sqrt{P\cdot(1-\rho)+N\cdot\rho}}$
TS_α	if $P=1\rightarrow\Theta^*\setminus\{0\}$ else $\rightarrow\{1\}$	$\frac{P\cdot\theta^*_{\text{opt}}}{P+N\cdot\theta^*_{\text{opt}}}$	$\frac{P\cdot(1-\rho)}{\rho\cdot N+P}$
$\text{G}^{(2)}_\alpha$	n/a	$\sqrt{\theta^*_{\text{opt}}\cdot(1-\theta^*_{\text{opt}})}$	$1-\rho$

5.4.5 Selecting θ^*_{opt} and Reverse Engineering α

The definitions of the performance metrics under the DS provided in Table 5.1 show that some metrics (e.g., J_α and BAcc_α) are independent of θ^*_{opt}, while other metrics (e.g., FM_α, F^β_α, or Acc_α) have a unique corresponding DD optimizer (θ^*_{opt}) as shown in Table 5.2, and there are metrics for which multiple θ^*_{opt} result in the DD baseline ($|\Theta^*_{\text{opt}}| > 1$). To avoid ambiguity regarding the choice of θ^*_{opt} when there are multiple options, we outline the selection criteria that should guide it. If one has obtained an empirical performance metric score $\overline{\mu}$, then θ^*_{opt} should be selected so that the least required indicator rate α to achieve this. The intuition behind minimizing α is that we want to maximize the utility of the DD baseline. If a higher DD baseline can be achieved without learning by selecting another θ^*_{opt}, then this θ^*_{opt} should be selected. If we can achieve a higher score without learning, the DD baseline difference should not be considered as learning. The DD baseline

parameter θ^*_{opt} should thus be selected to minimize α under the restriction that the constructed score μ_α equals the empirical gathered performance metric score $\overline{\mu}$.

Reverse engineering α with the selection criteria of θ^*_{opt} to obtain a metric score requires solving the following optimization problem for a selected performance metric μ:

$$\begin{aligned} \min \quad & \alpha, \\ \text{s.t.} \quad & \overline{\mu} = \mu_\alpha(\text{TP}_\alpha, \text{TN}_\alpha, \text{FP}_\alpha, \text{FN}_\alpha), \\ & \theta^*_{\text{opt}} \in \Theta^*_{\text{opt}}, \\ & \alpha \in [0, 1]. \end{aligned}$$

The solution to this problem specifies the weight α required to achieve the same realized performance score. Let us notate μ_α as a function $s = \mu_\alpha(\alpha, P, M, \theta^*_{\text{opt}}, \rho)$. Given that we are interested in α, we denote μ_α^{-1} as the inverse function of μ_α to $s : \alpha = \mu_\alpha^{-1}(s, P, M, \theta^*_{\text{opt}}, \rho)$. We search for the θ^*_{opt} by taking the derivative of μ_α^{-1} to θ^*_{opt} and then see what θ^*_{opt} results in the lowest α. Table 5.3 shows the DD strategies leading to the DD baseline and which θ^*_{opt} results in the lowest α for a subset of the selected performance metrics. The second column shows the θ^*_{opt} leading to the lowest $\alpha \in [0, 1]$, and the third column gives the DS definitions using these unique values, and we can exclude θ^*_{opt} from each function. Some metrics are independent of θ^*_{opt}, so multiple θ^*_{opt} lead to the same indicator score.

Table 5.3: Optimal θ^*_{opt} selection for performance metrics

Metric	$\arg\min_{\theta^*_{\text{opt}}}\{\mu_\alpha^{-1}\}$	μ_α **with** θ^*_{opt} **substituted**
PPV_α	$\{\frac{1}{M}\}$	$\frac{P}{M} \cdot \frac{\alpha\cdot(M\cdot(1-\rho)-1)+1}{\alpha\cdot(\rho\cdot(N-P)+P-1)+1}$
NPV_α	$\{\frac{M-1}{M}\}$	$\frac{N}{M} \cdot \frac{\alpha\cdot(M\cdot(1-\rho)-1)+1}{\alpha\cdot(\rho\cdot(P-N)+N-1)+1}$
Acc_α	if $P = N \rightarrow \Theta^*$ else $\rightarrow \{[P < \frac{M}{2}]\}$	$\frac{\alpha\cdot(\min\{P,N\}-M\cdot\rho)+\max\{P,N\}}{M}$
BAcc_α	Θ^*	$\alpha \cdot (1 - 2 \cdot \rho) \cdot \frac{1}{2} + \frac{1}{2}$
J_α	Θ^*	$\alpha \cdot (1 - 2 \cdot \rho)$
FM_α	$\{1\}$	$\sqrt{P} \cdot \frac{1-\alpha\cdot\rho}{\sqrt{\alpha\cdot(\rho\cdot(N-P)-N)+M}}$
F^β_α	$\{1\}$	$P \cdot (1 + \beta^2) \cdot \frac{1-\alpha\cdot\rho}{\alpha\cdot(\rho\cdot(N-P)-N)+M+P\cdot\beta^2}$
κ_α	if $P = N \rightarrow \Theta^*$ else $\rightarrow \{[P < \frac{M}{2}]\}$ if $P = 1$ and $\text{TS}_\alpha = \frac{1}{N} \rightarrow \Theta^* \setminus \{0\}$	$\frac{2\cdot P\cdot N\cdot\alpha\cdot(1-2\cdot\rho)}{\alpha\cdot(\rho\cdot(N-P)^2-(\min\{P,N\})^2+P\cdot N)+M\cdot\min\{P,N\}}$
TS_α	if $P = 1$ and $\text{TS}_\alpha > \frac{1}{N} \rightarrow \{\frac{1}{M}\}$ else $\rightarrow \{1\}$	See appendix (5.8.8)

5.4.6 Limitations for ρ

Ensuring an accurate approximation of the performance score of an 'optimal' classifier by the DO is crucial for two primary reasons. First, it establishes the target benchmark when searching for the 'optimal' model. Setting an incorrect target might result in prematurely concluding the search upon discovering a suboptimal model, or conversely, the search may persist indefinitely, even if finding the truly optimal model proves unattainable. Second, any inaccuracies in this approximation can potentially compromise our proposed indicator's desired properties, as detailed in Section 5.4.1.

We must set ρ at a correct value to guarantee that the DO performs better than the DD baseline. This can be accomplished by ensuring that μ_α exhibits a strictly increasing trend to α. Table 5.4 shows the derivative of each performance metric in α and specifies which ρ should be selected to satisfy both properties described in Section 5.4.1. For most performance metrics, $\rho < 0.5$ implies having a strictly increasing μ_α in α, while there are metrics where the limitation of ρ depends on P, M (and β for the F_β score). The last column specifies how ρ can be determined when the user knows the expected performance metric score of the 'optimal' model. This information is valuable if the user wants to estimate ρ on empirical estimates. The derivative of μ_α should be strictly increasing in α, and ρ should be smaller than the inverse DS function at the expected performance score of the 'optimal' classifier. In the next section, we will elaborate on categorizing performance metrics based on their second derivative.

Table 5.4: Limits for ρ to ensure indicator properties

Metric	**Derivative μ_α in α**	$\arg\max_\rho s.t. \frac{\partial \mu_\alpha}{\partial \alpha} > 0$	$\rho = \mu_1^{-1}(\Omega_\mu)$
Acc_α	$\frac{\min\{P,N\}}{M} - \rho$	$\min\{\frac{N}{M}, \frac{P}{M}\}$	$1 - \Omega_{\text{Acc}}$
BAcc_α	$\frac{1}{2} - \rho$	0.5	$1 - \Omega_{\text{Bacc}}$
J_α	$1 - 2 \cdot \rho$	0.5	$\frac{1}{2} - \frac{1}{2} \cdot \Omega_J$
PPV_α	$\frac{P \cdot N}{M} \cdot \frac{1-2\cdot\rho}{(\alpha\cdot(\rho\cdot(N-P)+P-1)+1)^2}$	0.5	$\frac{P\cdot(1-\Omega_{\text{PPV}})}{\Omega_{\text{PPV}}\cdot(N-P)+P}$
NPV_α	$\frac{P \cdot N}{M} \cdot \frac{1-2\cdot\rho}{(\alpha\cdot(\rho\cdot(P-N)+N-1)+1)^2}$	0.5	$\frac{N\cdot(1-\Omega_{\text{NPV}})}{\Omega_{\text{NPV}}\cdot(P-N)+N}$
F_α^β	$\frac{P\cdot(1+\beta^2)\cdot(N\cdot(1-2\cdot\rho)-\rho\cdot P\cdot\beta^2)}{(\alpha\cdot(\rho\cdot(N-P)-N)+M+P\cdot\beta^2)^2}$	$\frac{N}{2\cdot N+P\cdot\beta^2}$	$\frac{(1+\beta^2)\cdot P\cdot(1-\Omega_{F^\beta})}{\Omega_{F^\beta}\cdot(N-P)+P\cdot(1+\beta^2)}$
κ_α	$\frac{(1-2\cdot\rho)\cdot 2\cdot P\cdot N\cdot M\cdot\min\{P,N\}}{(\alpha\cdot(\rho\cdot(N-P)^2+P\cdot N-(\min\{P,N\})^2)+M\cdot\min\{P,N\})^2}$	0.5	$\frac{2\cdot P\cdot N\cdot(1-\Omega_\kappa)}{\Omega_\kappa\cdot(N-P)^2+4\cdot P\cdot N}$
FM_α	$\sqrt{P}\cdot\frac{\alpha\cdot\rho^2\cdot P+\alpha\cdot N\cdot\rho\cdot(1-\rho)+N\cdot(1-\rho)+\rho\cdot(P-2\cdot M)}{2\cdot(M-\alpha\cdot(N\cdot(1-\rho)+P\cdot\rho))^{\frac{3}{2}}}$	$\frac{N}{3\cdot N+P}$	See appendix (5.8.11)
MK_α	$(1-2\cdot\rho)\cdot P\cdot N\cdot\frac{M^2\cdot\theta^*_{\text{opt}}+\hat{P}^2-2\cdot\theta^*_{\text{opt}}\cdot M\cdot\hat{P}}{(\hat{P}\cdot\hat{N})^2}$	0.5	See appendix (5.8.13)
MCC_α	$(1-2\cdot\rho)\cdot\sqrt{P\cdot N}\cdot M\cdot\frac{\hat{P}\cdot(1-\theta^*_{\text{opt}})+\hat{N}\cdot\theta^*_{\text{opt}}}{(\hat{P}\cdot\hat{N})^{\frac{3}{2}}}$	0.5	See appendix (5.8.14)
TS_α	See appendix (5.8.8)	$\frac{N}{M+N}$	$\frac{P\cdot(1-\Omega_{\text{TS}})}{\Omega_{\text{TS}}\cdot N+P}$
$\text{G}_\alpha^{(2)}$	N.A.	0.5	$1 - \Omega_{\text{G}^{(2)}}$

5.5 Concavity Analysis of the DSPI

In the previous section, we demonstrated how the DS enables the transformation of metric scores into a uniform interval with reference points. We also examined the conditions under which the DS meets the two desirable properties of an indicator. However, the second property, a positive monotonic relationship between realized scores and the DSPI, raises questions about the concavity of the transformation, which is still unaddressed. We conduct a concavity analysis of the DSPI to explore this aspect further. Figure 5.1 shows three visualizations of the effect of α on μ_α for the selected metrics. Plot (a) shows that some performance metric combinations have identical starting points, which is the same DD baseline, but their concavity differs. In plot (b), we transformed each metric with a simple min-max transformation to the same starting and ending point to emphasize concavity differences. Plot (c) shows the effect of increasing ρ on the concavity of the performance metrics. The graphs collectively illustrate how the concavity of various evaluation metrics behaves under different transformations and parameter adjustments. It can be observed that the impact of increasing ρ is that the slope of the curves and the 'optimal' classifier bound decreases. The progression in these graphs highlights how the metrics' concavity is sensitive to the DS standardization and parameter variation.

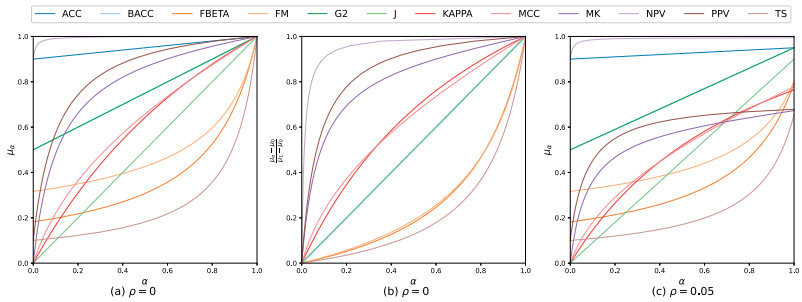


Figure 5.1: Visualizations of DSPI for $M = 100$ and $P = 10$

Figure 5.1 indicates concavity differences between the metrics. The critical question remains: what are the implications of the specific form of concavity, and do metrics consistently exhibit the same type of concavity? Let us use the following definition to categorize metrics under the DS on their second derivative.

Definition 1 (from [101])**.** *Suppose $f(x)$ is a function twice continuously differentiable on an interval I; then it is concave up, concave down, or linear, depending on whether for all $x \in I$ it holds that $\frac{\partial^2 f(x)}{\partial x^2} > 0$, $\frac{\partial^2 f(x)}{\partial x^2} < 0$, or $\frac{\partial^2 f(x)}{\partial x^2} = 0$, respectively.*

A concave-up relationship between performance metric scores and their corresponding DSPI value exhibits a pattern that mirrors the essence of the Pareto Principle (although not precisely): a disproportionate big DSPI increment is achieved for a relatively low metric score. Achieving an increment in DSPI becomes harder when increasing the performance score. For concave-down, it is exactly the other way around: a disproportionate large metric score is required for a relatively small increment in the DSPI. This concavity analysis helps determine how much better one model is over another when outperforming it in the performance metric score.

Table 5.5 shows the second derivative of each metric under the DS and indicates whether metrics are concave up/down or linear. The second derivative shows the curvature of each performance metric under the DS. All first derivatives are non-decreasing/positive, but the concavity of the performance metrics depends on the second derivative. The second derivative for the MK_α and MCC_α are derived, but whether the curvature is concave up/down or linear depends on the corresponding θ^*_{opt}. *Accuracy*, *balanced accuracy*, and the *J statistic* are always linear metrics in α. This means that the relative performance metric increase, starting from the DD baseline and ending at the DO, is one-to-one with an increment in how much a model has learned. There are only some situations where learning occurs linearly for *precision*, the *negative predictive value*, and the *kappa*. With this categorization of the performance metric, we can better interpret the relationship between increasing the performance metric score and the achieved quantification of how much a classifier has learned.

Table 5.5: Analysis of DSPI derivatives

Metric	Second derivative μ_α to α	Conc. Up?	Conc. Down?	Linear?
Acc_α	0	-	-	✓
BAcc_α	0	-	-	✓
J_α	0	-	-	✓
PPV_α	$\frac{\partial \text{PPV}_\alpha}{\partial \alpha} \cdot \frac{-2 \cdot (\rho \cdot (N-P) + P - 1)}{\hat{P}}$	-	✓	$P = N = 1$, $P = 1 \wedge \rho = 0$
NPV_α	$\frac{\partial \text{NPV}_\alpha}{\partial \alpha} \cdot \frac{-2 \cdot (\rho \cdot (P-N) + N - 1)}{\hat{N}}$	-	✓	$P = N = 1$, $N = 1 \wedge \rho = 0$
F^β_α	$\frac{\partial F^\beta_\alpha}{\partial \alpha} \cdot \frac{2 \cdot (N \cdot (1-\rho) + P \cdot \rho)}{P \cdot \beta^2 + \hat{P}}$	✓	-	-
κ_α	$\frac{\partial \kappa_\alpha}{\partial \alpha} \cdot \frac{-2 \cdot (\rho \cdot (N-P)^2 + P \cdot N - (\min\{P,N\})^2)}{\hat{P} \cdot N + \hat{N} \cdot P}$	-	✓	$P = N$
TS_α	$\frac{\partial \text{TS}_\alpha}{\partial \alpha} \cdot \frac{2 \cdot N \cdot (\theta^*_{\text{opt}} - \rho)}{\text{FP}_\alpha + P}$	✓	-	-
FM_α	$\frac{\partial \text{FM}_\alpha}{\partial \alpha} \cdot \frac{N \cdot (1-\rho) + \rho \cdot P}{2 \cdot \hat{P}}$	✓	-	-

5.6 Dutch Scaler in Practice

Now that all the DSPI's theoretical properties have been discussed, it is time to demonstrate its value in practice. First, we compare the DSPI with the ACCBAR and show their differences. Second, we demonstrate how the DS provides a means to compare classifiers on multiple selected metrics.

5.6.1 DSPI versus ACCBAR

Let us compare the DSPI with the values produced by the indicator ACCBAR. We used the same 28 studies to gather DSPI scores as were obtained for the ACCBAR [102]. The ACCBAR is derived by subtracting the baseline from the performance metric score. The resulting DSPI, ACCBAR, and scaled ACCBAR scores for the metric *accuracy* are shown in Figure 5.2. As shown in Table 5.5, *accuracy* is a measure for which learning is linearly in the increment of the performance metric scores when transformed by the DS. Therefore, when we min-max scale the ACCBAR scores with the baseline and the upper-performance metric score of the optimal model as reference points, it can be observed that these scaled scores are identical to the DSPI scores. The absolute values differ, but when scaled, they are identical. This means that for performance metrics that are linear in α, ACCBAR and DSPI assessments are identical.

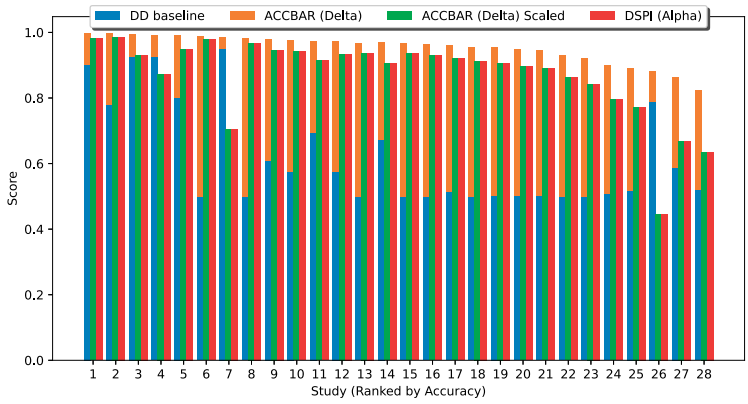


Figure 5.2: Performance indicator scores of DSPI and ACCBAR on *accuracy* for 28 different studies

Differences between the scaled ACCBAR and DSPI arrive when we consider performance metrics that, under the DS transformation, are not linear in α. Figure 5.3 shows computed ACCBAR, scaled ACCBAR, and DSPI scores for the F_1 score. Unlike Figure 5.2, only seven studies reported the F_1 score of their method, allowing us to compute the ACCBAR and DSPI on the F_1 scores only for these

studies. For this metric, it can be observed that the DSPI is higher than the scaled ACCBAR in all cases. This means the DSPI assesses that the model has learned more than the quantification computed with the ACCBAR. The F_β performance metric under the DS is a concave-up function, so the learning quantity assessed by the DS will always be higher for these kinds of metrics. When performance metrics are concave-down in the DS, the quantitative assessment of the prediction performance of a model will be lower than the ACCBAR will indicate. In general, the quality assessment of the DSPI, when compared to the scaled ACCBAR, will be more conservative when the metrics exhibit concave-down behavior. At the same time, a concave-up trend will result in a more favorable evaluation. Still, the DSPI scores have more practical meaning as they consider absolute prediction quality in contrast to the relative performance quantification by the ACCBAR.

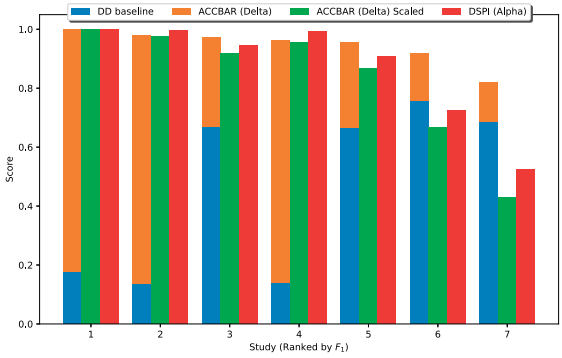


Figure 5.3: Performance indicator scores of DSPI and ACCBAR on the F_1 score for seven studies

5.6.2 Comparing Classifiers with the DSPI

As previously discussed, individual metrics often emphasize distinct aspects of the confusion matrix, making direct comparisons unfounded. For instance, a recall of 0.9 and a precision of 0.3 are meaningful when interpreted together but cannot be directly compared. The DSPI addresses this challenge by converting metrics into a unified scale, allowing for more coherent evaluations across classifiers. When applying this transformation, it makes more sense to average the standardized scores. This enables a more accurate and holistic assessment of a classifier's effectiveness. To demonstrate this, let us consider the results of a binary classification study. In Sidey-Gibbons and Sidey-Gibbons [103], three machine learning classifiers were used on the *Breast Cancer Wisconsin Diagnostic Data Set*, an often studied classification problem [104]. The task here is to predict whether tumors are benign or malignant.

Table 5.6 shows the base measure scores of these classifiers when evaluated on a test dataset, the corresponding metric scores, derived DSPI, and scaled ACCBAR scores. Three machine learning models, *generalised linear models* (GLM), *support vector machines* (SVM) and, *artificial neural networks* (ANN), were trained to classify $M = 227$ tumor instances into benign ($N = 150$) or malignant ($P = 77$). DSPI scores are more conservative in their prediction quality assessment than acquired metric scores, especially for PPV and NPV. One exception to this rule is the Threat Score, where the DSPI gives higher scores. Metrics concave-up under the DS are metrics where the DSPI gives higher scores than scaled ACCBAR scores, while metrics that are linear or concave-down under the DS will always have DSPI scores that are equal or more conservative. The scaled ACCBAR scores provide a performance indication of a classifier but fail to make valid comparisons between scores of different metrics. Looking at the average DSPI scores and metrics scores, it is valid to average the DSPI scores as they belong to the same uniform framework. A remark here is that it would only be appropriate to average metrics if they are all either naturally maximized or minimized. With these averaged DSPI scores, we can get a better holistic performance assessment.

Table 5.6: Comparison of metric (μ), DSPI (μ_α), and scaled ACCBAR (Δ) scores

	GLM			SVM			ANN		
Metric	$\overline{\mu}$	μ_α	Δ	$\overline{\mu}$	μ_α	Δ	$\overline{\mu}$	μ_α	Δ
TN	148	-	-	146	-	-	148	-	-
FN	10	-	-	5	-	-	11	-	-
FP	2	-	-	4	-	-	2	-	-
TP	67	-	-	72	-	-	66	-	-
PPV	0.971	0.221	0.956	0.947	0.130	0.920	0.971	0.218	0.955
NPV	0.937	0.028	0.813	0.967	0.058	0.902	0.931	0.025	0.796
ACC	0.947	0.844	0.844	0.960	0.883	0.883	0.943	0.831	0.831
BACC	0.928	0.857	0.857	0.954	0.908	0.908	0.922	0.844	0.844
FBETA	0.918	0.908	0.833	0.941	0.936	0.881	0.910	0.899	0.818
MCC	0.882	0.842	0.882	0.911	0.882	0.911	0.872	0.829	0.872
J	0.857	0.857	0.857	0.908	0.908	0.908	0.844	0.844	0.844
MK	0.908	0.806	0.908	0.914	0.821	0.914	0.901	0.792	0.901
KAPPA	0.879	0.846	0.879	0.911	0.886	0.911	0.869	0.833	0.869
FM	0.919	0.906	0.806	0.941	0.934	0.859	0.912	0.896	0.790
TS	0.848	0.908	0.770	0.889	0.936	0.832	0.835	0.899	0.751
Average	0.909	0.729	0.855	0.931	0.753	0.894	0.901	0.719	0.843

5.7 Discussion and Conclusion

In this chapter, we introduced the DS, a performance indicator to quantify how much a model actually has learned. Our indicator is an essential quantifier to interpret performance metric scores and assess the prediction quality of classifiers. Classifiers can be compared under the DS as it transforms metrics to a uniform reference framework, and we can determine how much better one model is. We showed how the DSPI scores can be derived for a set of commonly used performance metrics. We proposed two essential properties that an indicator should possess and show under which conditions a DS metric satisfies them to have a valid interpretation. If performance metrics under the DD baseline are linear in $\text{TP}_\theta^{\text{DD}}$, it is shown that the DD baseline equals the lower bound of the DS. If not, the quantification mechanism to contextualize performance metric scores will not (necessarily) work. For example, the DD baseline for the *G-mean 2 score* could be below the lower bound of the DS, making the alignment incorrect.

We proposed the DO as an imperfect oracle classifier to approximate the expected performance metric score of an 'optimal' model. This classifier makes a correct prediction with probability $1 - \rho$ independently of the instance. This ρ is a user-defined parameter. This novel classifier is independent of data and does not look at correlations between instances, so further research can propose better methods to improve this approximation. A better approximation could improve the quantification of how much a model has learned. Fortunately, the DS is a flexible framework in which the DO can easily be replaced with a more sophisticated model that approximates the performance of the 'optimal' model.

With our uniform reference framework, we categorized performance metrics on their concavity under the DS and showed how some metrics are more similar than others. The DS translates performance metrics scores into the same frame of reference to compare performance metrics based on their curvature of the second derivative. The first derivative is always positive as this is our desirable property, but this is not always true for the second derivative. With visualizations, we have shown the impact of increasing the performance metric score on the actual acquired prediction quality improvement. Performance metrics that are concave up, down, or linearly increasing indicate whether or not it is relatively easy/hard to increase the performance acquired by the model.

Although the primary focus of this chapter lies in quantifying the learning progress of binary classification models, the concept can be expanded to encompass models of various classifier types or those addressing different problem domains. For instance, one could examine multi-classification classifiers or explore regression problems. To integrate the DS framework into these problems, we first need

to define the expected performance of both a baseline and an 'optimal' model. In this context, the baseline model is input-independent, representing the expected score of a model that has not been learned. This approach allows us to determine the metric score for a non-learning model. The key question then becomes how to adjust the baseline measures in a multiclass classification problem or the output data points in a regression problem in the direction of the 'optimal' model to match the model's score. By doing so, we can quantify the extent to which models in these tasks have learned. In essence, developing indicators in these contexts enhances interpretability and improves the ability to assess the overall quality of models.

5.8 Appendix

This section contains the complete theoretical analysis used to gather the information presented in Sections 5.4.4, 5.4.5, 5.4.6, and 5.5. Each subsection is dedicated to one of the evaluation measures/metrics.

An overview of the entire appendix can be viewed in Table 5.7.

Table 5.7: Overview of the DSPI mathematical derivations

Measure	Base Measures	TPR/TNR	PPV	NPV	J	ACC	BAcc
Section	5.8.1	5.8.2	5.8.3	5.8.4	5.8.5	5.8.6	5.8.7
Measure	TS	κ	F_β	FM	$G^{(2)}$	MK	MCC
Section	5.8.8	5.8.9	5.8.10	5.8.11	5.8.12	5.8.13	5.8.14

5.8.1 Base Measures

The calculation of α for the base measures TP_α and TN_α is rather straightforward. The optimal DD baseline parameter for these base measures is $\theta^*_{\text{opt}} = 1$ and $\theta^*_{\text{opt}} = 0$, respectively. Hence, Equations (5.1) and (5.2) become:

$$\text{TP}_\alpha \coloneqq P \cdot (1 - \alpha \cdot \rho), \quad \text{TN}_\alpha \coloneqq N \cdot (1 - \alpha \cdot \rho).$$

We require $\dfrac{\partial \mu_\alpha}{\partial \alpha} > 0$ for $\forall \alpha \in [0, 1]$, but there is no value of $\rho \in [0, 1]$ to satisfy this requirement. This implies, furthermore, that $\mu_{\alpha_1} = \mu_{\alpha_2}$ for both TP_α and TN_α respectively for $\forall \alpha_1, \alpha_2 \in [0, 1]$ and these equations cannot be rewritten as a function of α. This makes sense because the DD baseline and DO are equal:

$\mathbb{E}\left[\mathrm{TP}_0^{\mathrm{DO}}\right] = \mathbb{E}\left[\mathrm{TP}_{\theta^*_{\mathrm{opt}}}^{\mathrm{DD}}\right] = P$, $\mathbb{E}\left[\mathrm{TN}_0^{\mathrm{DO}}\right] = \mathbb{E}\left[\mathrm{TN}_{\theta^*_{\mathrm{opt}}}^{\mathrm{DD}}\right] = N$ and every weight factor $\alpha \in [0, 1]$ will do.

5.8.2 True Positive Rate and True Negative Rate

The same reasoning holds for the *true positive rate* (TPR_α) and the *true negative rate* (TNR_α) as for TP_α and TN_α. The evaluation measures are defined as follows:

$$\mathrm{TPR}_\alpha \coloneqq \frac{\mathrm{TP}_\alpha}{P} = \alpha \cdot (1 - \rho - \theta^*_{\mathrm{opt}}) + \theta^*_{\mathrm{opt}}, \tag{5.7}$$

and

$$\mathrm{TNR}_\alpha \coloneqq \frac{\mathrm{TN}_\alpha}{N} = \alpha \cdot (\theta^*_{\mathrm{opt}} - \rho) + (1 - \theta^*_{\mathrm{opt}}), \tag{5.8}$$

by using Equations (5.1) and (5.2) under the DS, respectively. The corresponding optimal DD baseline strategies are $\theta^*_{\mathrm{opt}} = 1$ and $\theta^*_{\mathrm{opt}} = 0$ for TPR_α, TNR_α respectively.The following closed-form expressions are obtained by filling in these strategies into the definitions: $\mathrm{TPR}_\alpha = 1 - \alpha \cdot \rho$ and $\mathrm{TNR}_\alpha = 1 - \alpha \cdot \rho$.

Derivatives: The DS requires $\dfrac{\partial \mu_\alpha}{\partial \alpha} > 0$ for $\forall \alpha \in [0, 1]$ to have a strictly increasing performance score in α. The derivative of both definitions with respect to α is given by $-\rho$. Thus, the requirement for these metrics cannot be satisfied.

5.8.3 Positive Predictive Value

The *positive predictive value* (PPV_α) under the DS is defined as follows:

$$\mathrm{PPV}_\alpha \coloneqq \frac{\mathrm{TP}_\alpha}{\mathrm{TP}_\alpha + \mathrm{FP}_\alpha} = \frac{\alpha \cdot P \cdot (1 - \rho - \theta^*_{\mathrm{opt}}) + P \cdot \theta^*_{\mathrm{opt}}}{\alpha \cdot (\rho \cdot (N - P) + P - M \cdot \theta^*_{\mathrm{opt}}) + M \cdot \theta^*_{\mathrm{opt}}}, \tag{5.9}$$

by using Equations (5.1) and (5.5). The optimal DD baseline strategy for PPV_α is $\theta^*_{\mathrm{opt}} \in \Theta^* \setminus \{0\}$. Hence, there is not a unique θ^*_{opt} resulting in the DD baseline. We show in the determination of the DSPI from a realized PPV score later in this section that $\theta^*_{\mathrm{opt}} = \frac{1}{M}$ leads to the least required DO. By filling this θ^*_{opt} in the definition, the following PPV_α formulation is obtained

$$\mathrm{PPV}_\alpha \coloneqq \frac{\alpha \cdot P \cdot (M \cdot (1 - \rho) - 1) + P}{\alpha \cdot M \cdot (\rho \cdot (N - P) + P - 1) + M}. \tag{5.10}$$

From this definition, it can be derived that $\mathrm{PPV}_0 = \frac{P}{M}$, which equals the DD baseline, and $\mathrm{PPV}_1 = \frac{P \cdot (1-\rho)}{\rho \cdot (N-P)+P}$, which represents the highest performance value.

Derivatives: The DS requires $\frac{\partial \mu_\alpha}{\partial \alpha} > 0$ for $\forall \alpha \in [0,1]$ to have a strictly increasing performance score in α. The derivative of Equation (5.10) with respect to α is given by

$$\frac{\partial \text{PPV}_\alpha}{\partial \alpha} = \frac{P \cdot N}{M} \cdot \frac{1 - 2 \cdot \rho}{(\alpha \cdot (\rho \cdot (N - P) + P - 1) + 1)^2}.$$

It can be observed that the PPV_α is non-decreasing in α if $\rho < 0.5$. Now, we look at the second derivative to test if the function is concave up/down. It is given by

$$\frac{\partial^2 \text{PPV}_\alpha}{\partial \alpha^2} = -2 \cdot \frac{P \cdot N \cdot (1 - 2 \cdot \rho)}{M} \cdot \frac{\rho \cdot (N - P) + P - 1}{(\alpha \cdot (\rho \cdot (N - P) + P - 1) + 1)^3}.$$

It can be observed that this function is concave down as the second derivative is, but linear in two scenarios: (1) $\rho = 0$ and $P = 1$; and (2) $N = P = 1$.

Upper Bound: Let us denote Ω_{PPV} as the expected PPV score of the 'optimal' model and assume its value is known. To satisfy the desirable property that this value equals μ_1, ρ should be

$$\rho = \frac{P \cdot (1 - \Omega_{\text{PPV}})}{\Omega_{\text{PPV}} \cdot (N - P) + P}. \tag{5.11}$$

Determining the DSPI: Let us show the optimal θ^*_{opt} for the PPV_α. Solving Equation (5.9) for α yields

$$\alpha = \frac{\theta^*_{\text{opt}} \cdot (\overline{\text{PPV}} \cdot M - P)}{\theta^*_{\text{opt}} \cdot (M \cdot \overline{\text{PPV}} - P) + P \cdot (1 - \overline{\text{PPV}}) + \rho \cdot (\overline{\text{PPV}} \cdot (P - N) - P)}, \tag{5.12}$$

with $\overline{\text{PPV}}$ being the empirical performance score. Remember that α quantifies how much the model has learned. In the case of multiple possible values of α, we want to choose the smallest value to know how much is learned at least. To see whether α is decreasing or increasing in θ^*_{opt}, we calculate the derivative of the function $f : \left[\frac{1}{2M}, 1\right] \to [0,1]$. It is given by

$$f(\theta) = \frac{\theta \cdot (M \cdot \overline{\text{PPV}} - P)}{\theta \cdot (M \cdot \overline{\text{PPV}} - P) + P \cdot (1 - \overline{\text{PPV}}) + \rho \cdot (\overline{\text{PPV}} \cdot (P - N) - P)}.$$

This function is the continuous version of Equation (5.12), note that $\alpha = f(\theta^*_{\text{opt}})$. The derivative of this inverse function is given by

$$\frac{\partial f(\theta)}{\partial \theta} = \frac{(M \cdot \overline{\text{PPV}} - P) \cdot (P \cdot (1 - \overline{\text{PPV}}) + \rho \cdot (\overline{\text{PPV}} \cdot (P - N) - P))}{\left(\theta \cdot (M \cdot \overline{\text{PPV}} - P) + P \cdot (1 - \overline{\text{PPV}}) + \rho \cdot (\overline{\text{PPV}} \cdot (P - N) - P)\right)^2}.$$

As we assumed that $\frac{P}{M} = \mu_0 \leq \overline{\text{PPV}}$, the equality in Equation (5.11) and the denominator is always non-negative, the sign of $\frac{\partial f(\theta)}{\partial \theta}$ is always positive thus the function is increasing in the interval Θ^*_μ. This implies that the minimal value of α is reached for the smallest possible θ^*_{opt}. Since $\theta^*_{\text{opt}} \in \Theta^* \setminus \{0\}$, the smallest possible value is $\frac{1}{M}$. Plugging this in into Equation (5.12) gives

$$\alpha = \frac{M \cdot \overline{\text{PPV}} - P}{M \cdot \overline{\text{PPV}} - P + P \cdot M \cdot (1 - \overline{\text{PPV}}) + \rho \cdot M \cdot (\overline{\text{PPV}} \cdot (P - N) - P)}.$$

5

Scaling the DSPI: Equation (5.10) can be analyzed in its behavior by scaling the scores with a min-max scaler between μ_0 and μ_1 with the PP ($\rho = 0$). This results in

$$\frac{\text{PPV}_\alpha|\rho = 0 - \mu_0}{\mu_1|\rho = 0 - \mu_0} = \frac{\frac{\alpha \cdot P \cdot (M-1)+P}{\alpha \cdot M \cdot (P-1)+M} - \frac{P}{M}}{1 - \frac{P}{M}} = \alpha \cdot \frac{P}{\alpha \cdot P + (1 - \alpha)}.$$

5.8.4 Negative Predictive Value

The same steps for the PPV are applied to the *negative predictive value* (NPV_α). It is defined as follows:

$$\begin{aligned} \text{NPV}_\alpha &:= \frac{\text{TN}_\alpha}{\text{TN}_\alpha + \text{FN}_\alpha} \\ &= \frac{\alpha \cdot N \cdot (\theta^*_{\text{opt}} - \rho) + N \cdot (1 - \theta^*_{\text{opt}})}{\alpha \cdot (\rho \cdot (P - N) + M \cdot \theta^*_{\text{opt}} - P) + M \cdot (1 - \theta^*_{\text{opt}})}, \end{aligned} \tag{5.13}$$

by using Equations (5.2) and (5.6). The optimal DD strategy for NPV_α is $\theta^*_{\text{opt}} \in \Theta^* \setminus \{1\}$. Thus, there is not a unique θ^*_{opt} resulting in the DD baseline. We will show in the determination of the DSPI from a realized NPV score later in this section that $\theta^*_{\text{opt}} = \frac{M-1}{M}$ leads to the lowest DSPI. By filling this θ^*_{opt} in the definition, the following PPV_α formulation is obtained

$$\text{NPV}_\alpha = \frac{\alpha \cdot N \cdot (\frac{M-1}{M} - \rho) + \frac{N}{M}}{\alpha \cdot (\rho \cdot (P - N) + N - 1) + 1}.$$

From this definition, it can be derived that $\text{NPV}_0 = \frac{N}{M}$, which equals the DD baseline, and $\text{NPV}_1 = \frac{N \cdot (1-\rho)}{\rho \cdot (P-N)+N}$, indicating the highest performance value.

Derivatives: The DS requires $\frac{\partial \mu_\alpha}{\partial \alpha} > 0$ for $\forall \alpha \in [0, 1]$ to have a strictly increasing performance score in α. The derivative with respect to α of the NPV under

the DS is given by

$$\frac{\partial \text{NPV}_\alpha}{\partial \alpha} = \frac{P \cdot N}{M} \cdot \frac{1 - 2 \cdot \rho}{(\alpha \cdot (\rho \cdot (P - N) + N - 1) + 1)^2}.$$

This shows that the NPV_α is non-decreasing in α if $\rho < 0.5$. Let us look at the second derivative, whether the function is concave up/down. The second derivative is given by

$$\frac{\partial^2 \text{NPV}_\alpha}{\partial \alpha^2} = -2 \cdot \frac{P \cdot N \cdot (1 - 2 \cdot \rho)}{M} \cdot \frac{\rho \cdot (P - N) + N - 1}{(\alpha \cdot (\rho \cdot (P - N) + N - 1) + 1)^3}.$$

It can be observed that this function is concave down but linear in two scenarios: (1) $\rho = 0$ and $N = 1$; and (2) $N = P = 1$.

Upper Bound: Let us denote Ω_{NPV} as the expected NPV score of the 'optimal' model, and its value is known. To satisfy the desirable property that this value equals μ_1, ρ should be

$$\rho = \frac{N \cdot (1 - \Omega_{\text{NPV}})}{\Omega_{\text{NPV}} \cdot (P - N) + N}. \tag{5.14}$$

Determining the DSPI: Let us show the optimal θ^*_{opt} for the NPV_α. Writing Equation (5.13) in α yields

$$\alpha = \frac{\theta^*_{\text{opt}} \cdot (\overline{\text{NPV}} \cdot M - N) + N - \overline{\text{NPV}} \cdot M}{\theta^*_{\text{opt}} \cdot (M \cdot \overline{\text{NPV}} - N) - \overline{\text{NPV}} \cdot P + \rho \cdot (\overline{\text{NPV}} \cdot (P - N) + N)}, \tag{5.15}$$

with $\overline{\text{NPV}}$ the realized value of NPV_α. Note that α quantifies how much the model has learned. In the case of multiple possible values of α, we want to choose the smallest value to know how much is learned at least. To see whether α is decreasing or increasing in θ^*_{opt}, we determine the derivative of the function $f : [0, 1 - \frac{1}{2M}) \to [0, 1]$ given by

$$f(\theta) = \frac{\theta \cdot (\overline{\text{NPV}} \cdot M - N) + N - \overline{\text{NPV}} \cdot M}{\theta \cdot (M \cdot \overline{\text{NPV}} - N) - \overline{\text{NPV}} \cdot P + \rho \cdot (\overline{\text{NPV}} \cdot (P - N) + N)}.$$

The function f is the continuous version of Equation (5.15): $\alpha = f(\theta^*_{\text{opt}})$. The derivative of f w.r.t. θ is given by

$$\frac{\partial f(\theta)}{\partial \theta} = \frac{\left(\overline{\text{NPV}} \cdot M - N\right) \cdot \left(\overline{\text{NPV}} \cdot N - N + \rho \cdot (\overline{\text{NPV}} \cdot (P - N) + N)\right)}{\left(\theta^*_{\text{opt}} \cdot (M \cdot \overline{\text{NPV}} - N) - \overline{\text{NPV}} \cdot P + \rho \cdot (\overline{\text{NPV}} \cdot (P - N) + N)\right)^2}.$$

The denominator is always non-negative, just as $\overline{\text{NPV}} > \frac{N}{M}$, and by Equation (5.14), the numerator will always be negative. Thus, the sign of $\frac{\partial f(\theta)}{\partial \theta}$ is

always non-positive; hence, f is strictly decreasing. Therefore, the minimal value of α is achieved for the largest possible θ^*_{opt}. Since $\theta^*_{\text{opt}} \in \Theta^* \backslash \{1\}$, the largest possible value is $\frac{M-1}{M}$. Substituting this value into Equation (5.15) gives

$$\alpha = \frac{N - \overline{\text{NPV}} \cdot M}{\overline{\text{NPV}} \cdot M \cdot (N-1) - N \cdot (M-1) + \rho \cdot (\overline{\text{NPV}} \cdot (P-N) + N) \cdot M}.$$

Scaling the DSPI: Equation (5.13) can be analyzed in its behavior by scaling the scores with a min-max scaler between μ_0 and μ_1 with the PP ($\rho = 0$). This gives us

$$\frac{\text{NPV}_\alpha | \rho = 0 - \mu_0}{\mu_1 | \rho = 0 - \mu_0} = \frac{\frac{\alpha \cdot N \cdot (M-1) + N}{\alpha \cdot M \cdot (N-1) + M} - \frac{N}{M}}{1 - \frac{N}{M}} = \alpha \cdot \frac{N}{\alpha \cdot N + (1-\alpha)}.$$

5.8.5 Youden's J Statistic

Youden's J statistic (J_α) under the DS is defined as follows:

$$\begin{aligned} J_\alpha &\coloneqq \text{TPR}_\alpha + \text{TNR}_\alpha - 1 \\ &= \alpha \cdot (1 - \rho - \theta^*_{\text{opt}}) + \theta^*_{\text{opt}} + \alpha \cdot (\theta^*_{\text{opt}} - \rho) + (1 - \theta^*_{\text{opt}}) - 1 \\ &= \alpha \cdot (1 - 2 \cdot \rho), \end{aligned} \tag{5.16}$$

by using Equations (5.7) and (5.8). The optimal DD strategy for NPV_α is $\theta^*_{\text{opt}} \in \Theta^*$, but the definition of the J_α statistic implies that the indicator performance value is independent of θ^*_{opt}, so any θ^*_{opt} would suffice. From the definition, it can be derived that $\text{J}_0 = 0$, which equals the DD baseline, and $\text{J}_1 = 1 - 2 \cdot \rho$, indicating the highest performance value.

Derivatives: The DS requires $\frac{\partial \mu_\alpha}{\partial \alpha} > 0$ for $\forall \alpha \in [0,1]$ to have a strictly increasing performance score in α. The derivative with respect to α of the J statistic under the DS is given by

$$\frac{\partial J_\alpha}{\partial \alpha} = 1 - 2 \cdot \rho.$$

The J statistic is thus non-decreasing in α when $\rho < 0.5$. It can be easily observed that the second derivative with respect to α of this function gives 0, indicating that it is linear on $\alpha \in [0,1]$.

Upper Bound: Let us denote Ω_J as the expected J score of the 'optimal' model, and its value is known. To satisfy the desirable property that this value equals μ_1, ρ should be

$$\rho = \frac{1 - \Omega_J}{2}.$$

Determining the DSPI: Suppose we have a realized $\overline{J}$ at our disposal. The DSPI can be derived by taking the inverse function of Equation (5.16), as we already excluded θ^*_{opt} from the equation. The DSPI is thus determined by

$$\alpha = \frac{\overline{J}}{1 - 2 \cdot \rho}.$$

Scaling the DSPI: Equation (5.16) can be analyzed in its behavior by scaling the scores with a min-max scaler between μ_0 and μ_1 with the PP ($\rho = 0$). This gives us

$$\frac{J_\alpha|\rho = 0 - \mu_0}{\mu_1|\rho = 0 - \mu_0} = \frac{\alpha - 0}{1 - 0} = \alpha.$$

This shows that the scaled DS function is linear in α.

5.8.6 Accuracy

The *accuracy* (Acc_α) under the DS is defined as follows:

$$\begin{aligned}\text{Acc}_\alpha &:= \frac{\text{TP}_\alpha + \text{TN}_\alpha}{M} \\ &= \frac{\alpha \cdot P \cdot (1 - \rho - \theta^*_{\text{opt}}) + P \cdot \theta^*_{\text{opt}} + \alpha \cdot N \cdot (\theta^*_{\text{opt}} - \rho) + N \cdot (1 - \theta^*_{\text{opt}})}{M},\end{aligned}$$

by using Equations (5.1) and (5.2). The optimal DD strategy (setting θ^*_{opt} to derive the DD baseline) depends on the P and N ratio:

$$\theta^*_{\text{opt}} \in \begin{cases} \{0\} & \text{if } P < N, \\ \{1\} & \text{if } P > N, \\ \Theta^* & \text{if } P = N. \end{cases}$$

By filling in this strategy setting into the definition of Acc_α, the following closed-form expression is obtained

$$\text{Acc}_\alpha = \frac{\alpha \cdot (\min\{P, N\} - M \cdot \rho) + \max\{P, N\}}{M}. \tag{5.17}$$

If we look at the boundaries of this function, it can be observed that $\text{Acc}_0 = \frac{\max\{P,N\}}{M}$, which equals the DD baseline, and $\text{Acc}_1 = 1 - \rho$ indicates the highest performance value.

Derivatives: The DS requires $\frac{\partial \mu_\alpha}{\partial \alpha} > 0$ for $\forall \alpha \in [0, 1]$ to have a strictly increasing performance score in α. The derivative of Acc_α (using Equation (5.17)) with respect to α is given by

$$\frac{\partial \text{Acc}_\alpha}{\partial \alpha} = \frac{\min\{P, N\}}{M} - \rho.$$

Thus, to satisfy the mentioned requirement, it should hold that $\rho < \frac{\min\{P,N\}}{M}$. It can be easily observed that the second derivative with respect to α of Acc_α is 0, indicating that the function is linear on $\alpha \in [0, 1]$.

Upper Bound: Let us denote Ω_{Acc} as the expected Acc score of the 'optimal' model, and its value is known. To satisfy the desirable property that this value equals μ_1, ρ should be

$$\rho = 1 - \Omega_{\text{Acc}}.$$

Determining the DSPI: Suppose we have a realized $\overline{\text{Acc}}$ at our disposal. The DSPI is derived by inverting Equation (5.17) in α. As only one unique θ^*_{opt} leads to the DD baseline, there is no need to optimize θ^*_{opt}, leading to the lowest DSPI score. Thus, the DSPI is determined by

$$\alpha = \frac{M \cdot \overline{\text{Acc}} - \max\{P, N\}}{\min\{P, N\} - M \cdot \rho}.$$

Scaling the DS: Equation (5.17) can be analyzed in its behavior by scaling the scores with a min-max scaler between μ_0 and μ_1 with the PP ($\rho = 0$). This gives us

$$\frac{\text{Acc}_\alpha|\rho = 0 - \mu_0}{\mu_1|\rho = 0 - \mu_0} = \frac{\frac{\alpha \cdot (\min\{P,N\}) + \max\{P,N\}}{M} - \frac{\max\{P,N\}}{M}}{1 - \frac{\max\{P,N\}}{M}} = \alpha.$$

This shows that the scaled DSPI function is linear in α.

5.8.7 Balanced Accuracy

The *balanced accuracy* (BAcc_α) under the DS is defined as follows:

$$\begin{aligned}
\text{BAcc}_\alpha &:= \frac{1}{2}\left(\text{TPR}_\alpha + \text{TNR}_\alpha\right) \\
&= \frac{1}{2}\left(\alpha \cdot (1 - \rho - \theta^*_{\text{opt}}) + \theta^*_{\text{opt}} + \alpha \cdot (\theta^*_{\text{opt}} - \rho) + (1 - \theta^*_{\text{opt}})\right) \\
&= \alpha \cdot \frac{(1 - 2 \cdot \rho)}{2} + \frac{1}{2}, \qquad (5.18)
\end{aligned}$$

by using Equations (5.7) and (5.8). The optimal DD strategy for the BAcc is $\theta^*_{\text{opt}} \in \Theta^*$. Fortunately, BAcc_α is independent of θ^*_{opt}, so any θ^*_{opt} in this set would give the same result, and no optimization of θ^*_{opt} is required. If we look at the bounds of the BAcc_α, it can be observed that $\text{BAcc}_0 = \frac{1}{2}$, which equals the DD baseline, and $\text{BAcc}_1 = 1 - \rho$ indicates the highest performance value.

Derivatives: The DS requires $\frac{\partial \mu_\alpha}{\partial \alpha} > 0$ for $\forall \alpha \in [0,1]$ to have a strictly increasing performance score in α. The derivative of BAcc_α with respect to α is given by

$$\frac{\partial \text{BAcc}_\alpha}{\partial \alpha} = \frac{1}{2} - \rho.$$

Thus, $\rho < 0.5$ satisfies the requirement of a strictly increasing DSPI. The second derivative, with respect to α, gives 0, indicating that the function is linear on $\alpha \in [0,1]$.

Upper Bound: Let us denote Ω_{BAcc} as the expected BAcc score of the 'optimal' model and its value is known. To satisfy the desirable property that this value equals μ_1, ρ should be

$$\rho = 1 - \Omega_{\text{BAcc}}.$$

Determining the DSPI: Now, suppose we have a realized $\overline{\text{BAcc}}$ at our disposal. The DSPI can be derived by taking the inverse function of Equation (5.18), as we already excluded θ^*_{opt} from the equation. The DSPI is thus determined by

$$\alpha = \frac{2 \cdot \overline{\text{BAcc}} - 1}{1 - 2 \cdot \rho}.$$

Scaling the DS: Equation (5.18) can be analyzed in its behavior by scaling the scores with a min-max scaler between μ_0 and μ_1 with the PP ($\rho = 0$). This gives us

$$\frac{\text{BAcc}_\alpha | \rho = 0 - \mu_0}{\mu_1 | \rho = 0 - \mu_0} = \frac{\alpha \cdot \frac{1}{2} + \frac{1}{2} - \frac{1}{2}}{1 - \frac{1}{2}} = \alpha.$$

This shows that the scaled DS is linear in α.

5.8.8 Threat Score

The *threat score* (TS_α) under the DS is defined as follows:

$$\text{TS}_\alpha \coloneqq \frac{\text{TP}_\alpha}{\text{TP}_\alpha + \text{FP}_\alpha + \text{FN}_\alpha} = \frac{\alpha \cdot P \cdot (1 - \rho - \theta^*_{\text{opt}}) + P \cdot \theta^*_{\text{opt}}}{P + \alpha \cdot N \cdot (\rho - \theta^*_{\text{opt}}) + N \cdot \theta^*_{\text{opt}}}, \tag{5.19}$$

by using Equations (5.1), (5.2) and (5.4). The optimal DD strategy for TS_α is given by

$$\theta^*_{\text{opt}} \in \begin{cases} \Theta^* \setminus \{0\} & \text{if } P = 1, \\ \{1\} & \text{if } P > 1. \end{cases}$$

This gives us the following definition with the corresponding strategy:

$$\mathrm{TS}_\alpha := \begin{cases} \frac{\alpha\cdot(1-\rho-\theta^*_{\text{opt}})+\theta^*_{\text{opt}}}{1+\alpha\cdot N\cdot(\rho-\theta^*_{\text{opt}})+N\cdot\theta^*_{\text{opt}}} & \text{if } P=1, \\ P\cdot\frac{1-\alpha\cdot\rho}{M-N\cdot\alpha\cdot(1-\rho)} & \text{if } P>1. \end{cases}$$

Unfortunately, there is not a unique θ^*_{opt} for $P=1$ resulting in the DD baseline. We will show in the determination of the DSPI from a realized TS score later in this section that

$$\theta^*_{\text{opt}} \in \begin{cases} \Theta^* \setminus \{0\} & \text{if } P=1 \text{ and } \overline{\mathrm{TS}}=\frac{1}{N}, \\ \{\frac{1}{M}\} & \text{if } P=1 \text{ and } \overline{\mathrm{TS}}>\frac{1}{N}, \\ \{1\} & \text{if } P>1 \text{ or } P=1\wedge\overline{\mathrm{TS}}<\frac{1}{N}, \end{cases}$$

leads to the lowest DSPI. For $P=1$, we can observe that $\mathrm{TS}_\alpha=\frac{1}{N}$ is the inflection point for a different θ^*_{opt}. The α resulting in this value is $\frac{1}{N\cdot(1-2\cdot\rho)}$ for the TS with $P=1$. By filling this θ^*_{opt} in the definition, the following TS_α formulation is obtained

$$\mathrm{TS}_\alpha = \begin{cases} \frac{\alpha\cdot(N-M\cdot\rho)+1}{M+N-\alpha\cdot N\cdot(1-M\cdot\rho)} & \text{if } \{P=1\wedge\frac{1}{N\cdot(1-2\cdot\rho)}\le\alpha\le 1\}, \\ P\cdot\frac{1-\alpha\cdot\rho}{M-N\cdot\alpha\cdot(1-\rho)} & \text{if } \{P>1\} \text{ or } \{P=1\wedge 0\le\alpha\le\frac{1}{N\cdot(1-2\cdot\rho)}\}. \end{cases}$$

If we look at the boundaries of this function, it can be observed that $\mathrm{TS}_0=\frac{P}{M}$, which equals the DD baseline, and $\mathrm{TS}_1=\frac{P\cdot(1-\rho)}{P+N\cdot\rho}$ indicates the highest performance value.

Derivatives: To satisfy the assumption that $\frac{\partial\mu_\alpha}{\partial\alpha}>0$ for $\forall\alpha\in[0,1]$, we should check for which ρ this is the case. Its derivative is given by

$$\frac{\partial\mathrm{TS}_\alpha}{\partial\alpha} = \begin{cases} \frac{1-\rho\cdot(2\cdot N\cdot\theta^*_{\text{opt}}+1)+(N-1)\cdot\theta^*_{\text{opt}}}{(1+\alpha\cdot N\cdot(\rho-\theta^*_{\text{opt}})+N\cdot\theta^*_{\text{opt}})^2} & \text{if } \{P=1\}, \\ P\cdot\frac{N-\rho\cdot(M+N)}{(M-N\cdot\alpha\cdot(1-\rho))^2} & \text{if } \{P>1\}. \end{cases}$$

When $P>1$, we see that $\rho<\frac{N}{N+M}$ should hold to have a strictly increasing score. When $P=1$, it should hold that

$$\rho < \frac{1+(N-1)\cdot\theta^*_{\text{opt}}}{2\cdot N\cdot\theta^*_{\text{opt}}+1}.$$

We have multiple strategies for θ^*_{opt}, so substituting them gives us

$$\rho < \begin{cases} \frac{N}{N+M} & \text{if } \{P>1\}, \\ \frac{N}{N+M} & \text{if } \{P=1\cap\theta^*_{\text{opt}}=1\}, \\ \frac{2\cdot N}{2\cdot N+M} & \text{if } \{P=1\cap\theta^*_{\text{opt}}=\frac{1}{M}\}. \end{cases}$$

Combining everything gives the restriction $\rho < \frac{N}{N+M}$. Let us now look at the second derivative, which is given by

$$\frac{\partial^2 \text{TS}_\alpha}{\partial \alpha^2} = \begin{cases} 2 \cdot N \cdot \frac{(\theta^*_{\text{opt}} - \rho) \cdot (1 - \rho \cdot (2 \cdot N \cdot \theta^*_{\text{opt}} + 1) + (N-1) \cdot \theta^*_{\text{opt}})}{(1 + \alpha \cdot N \cdot (\rho - \theta^*_{\text{opt}}) + N \cdot \theta^*_{\text{opt}})^3} & \text{if } \{P = 1\}, \\ 2 \cdot N \cdot P \cdot \frac{(1-\rho) \cdot (N \cdot (1 - 2 \cdot \rho) - P \cdot \rho)}{(N \cdot (1-\alpha) + P + N \cdot \alpha \cdot \rho))^3} & \text{if } \{P > 1\}. \end{cases}$$

We can say that on $\{P > 1\}$, the function is concave up due to the restriction of ρ. This also holds for $\{P = 1\}$ as $\theta^*_{\text{opt}} > \rho$ in all situations.

Upper Bound: Let us denote Ω_{TS} as the expected TS score of the 'optimal' model, and its value is known. To satisfy the desirable property that this value equals μ_1, ρ should be

$$\rho = \frac{P \cdot (1 - \Omega_{\text{TS}})}{P + N \cdot \Omega_{\text{TS}}}. \tag{5.20}$$

Determining the DSPI: Let us show the optimal θ^*_{opt} for the TS_α. Solving Equation (5.19) for α yields

$$\alpha = \frac{\theta^*_{\text{opt}} \cdot (\overline{\text{TS}} \cdot N - P) + \overline{\text{TS}} \cdot P}{\theta^*_{\text{opt}} \cdot (\overline{\text{TS}} \cdot N - P) - \overline{\text{TS}} \cdot N \cdot \rho + P \cdot (1 - \rho)}, \tag{5.21}$$

with $\overline{\text{TS}}$ the realized value of TS. Filling in the θ^*_{opt} resulting in the DD baseline results in

$$\alpha = \begin{cases} \frac{\theta^*_{\text{opt}} \cdot (\overline{\text{TS}} \cdot N - 1) + \overline{\text{TS}}}{\theta^*_{\text{opt}} \cdot (\overline{\text{TS}} \cdot N - 1) - \overline{\text{TS}} \cdot N \cdot \rho + (1 - \rho)} & \text{if } P = 1, \\ \frac{\overline{\text{TS}} \cdot M - P}{\overline{\text{TS}} \cdot N \cdot (1 - \rho) - P \cdot \rho} & \text{if } P > 1. \end{cases}$$

To see whether α is decreasing or increasing in θ^*_{opt} for the case that $P = 1$, we consider the function $f : [\frac{1}{2M}, 1] \to [0, 1]$ given by

$$f(\theta) = \frac{\theta \cdot (\overline{\text{TS}} \cdot N - 1) + \overline{\text{TS}}}{\theta \cdot (\overline{\text{TS}} \cdot N - 1) - \overline{\text{TS}} \cdot N \cdot \rho + (1 - \rho)}.$$

This function is the continuous version of Equation (5.21) with $P = 1$, note that $\alpha = f(\theta^*_{\text{opt}})$. The derivative of this function with respect to θ^*_{opt} is given by

$$\frac{\partial f(\theta)}{\partial \theta} = \frac{(\overline{\text{TS}} \cdot N - 1) \cdot (1 - \overline{\text{TS}} - \rho - \overline{\text{TS}} \cdot N \cdot \rho))}{(\theta \cdot (\overline{\text{TS}} \cdot N - 1) - \overline{\text{TS}} \cdot N \cdot \rho + (1 - \rho))^2}.$$

This function is either increasing in θ^*_{opt}, decreasing or stationary depending on $\overline{\text{TS}} \cdot N - 1$ as the denominator is always positive and the second term in the numerator is always non-negative due to Equation (5.20). So, (1) if $\overline{\text{TS}} < \frac{1}{N}$,

then the function is decreasing in θ^*_{opt} and we must select the highest $\theta^*_{\text{opt}} = 1$, (2) if $\overline{\text{TS}} > \frac{1}{N}$, the function is increasing in θ^*_{opt} and we select the lowest value $\theta^*_{\text{opt}} = \frac{1}{M}$, and (3) when $\overline{\text{TS}} = \frac{1}{N}$, it will always result in the same value and thus $\theta^*_{\text{opt}} \in \Theta^* \setminus \{0\}$. Plugging in these optimizers for θ^*_{opt} gives us

$$\alpha = \begin{cases} \frac{\overline{\text{TS}} \cdot (M+N) - 1}{\overline{\text{TS}} \cdot N - M \cdot \overline{\text{TS}} \cdot N \cdot \rho - M \cdot \rho + N} & \text{if } \{P = 1 \wedge \overline{\text{TS}} \geq \frac{1}{N}\}, \\ \frac{\overline{\text{TS}} \cdot M - P}{\overline{\text{TS}} \cdot N \cdot (1-\rho) - P \cdot \rho} & \text{if } \{P > 1\} \text{ or } \{P = 1 \wedge \overline{\text{TS}} \leq \frac{1}{N}\}. \end{cases}$$

Scaling the DSPI: Equation (5.10) can be analyzed in its behavior by scaling the scores with a min-max scaler between μ_0 and μ_1 with the PP ($\rho = 0$). This gives us

$$\frac{\text{TS}_\alpha | \rho = 0 - \mu_0}{\mu_1 | \rho = 0 - \mu_0} = \begin{cases} M \cdot \frac{\alpha \cdot (M+1) - 1}{M + N - \alpha \cdot N} & \text{if } \{P = 1 \wedge \frac{1}{N \cdot (1-2 \cdot \rho)} \leq \alpha \leq 1\}, \\ \alpha \cdot \frac{P}{M - N \cdot \alpha} & \text{if } \{P > 1\} \text{ or } \{P = 1 \wedge 0 \leq \alpha \leq \frac{1}{N \cdot (1-2 \cdot \rho)}\}. \end{cases}$$

5.8.9 Cohen's CKa

Cohen's kappa (κ_α) under the DS is defined as follows:

$$\kappa_\alpha \coloneqq \frac{\text{P}_o - \text{P}_e}{1 - \text{P}_e}, \text{ with } \text{P}_o = \text{Acc} = \frac{\text{TP}_\alpha + \text{TN}_\alpha}{M}, \text{P}_e = \frac{\hat{P} \cdot P + \hat{N} \cdot N}{M^2}.$$

Combining all components results in the following definition:

$$\kappa_\alpha \coloneqq 2 \cdot \frac{\text{TP}_\alpha \cdot \text{TN}_\alpha - \text{FN}_\alpha \cdot \text{FP}_\alpha}{\hat{P} \cdot N + \hat{N} \cdot P} = \alpha \cdot (1 - 2 \cdot \rho) \cdot \frac{2 \cdot P \cdot N}{\hat{P} \cdot N + \hat{N} \cdot P}.$$

Writing this function in terms of the DSPI (α) results in

$$\kappa_\alpha \coloneqq \frac{\alpha \cdot (1 - 2 \cdot \rho) \cdot 2 \cdot P \cdot N}{\alpha \cdot (N - P) \cdot (\rho \cdot (N - P) + P - M \cdot \theta^*_{\text{opt}}) + M \cdot (P + (N - P) \cdot \theta^*_{\text{opt}})}. \tag{5.22}$$

The optimal DD strategy for κ_α is $\theta^*_{\text{opt}} = \Theta^*$, except when $P = M$, then we have $\theta^*_{\text{opt}} = \Theta^* \setminus \{1\}$. However, if $P = M$, then $N = 0$ and thus $\kappa_\alpha = 0$. We will show, later in this section, that the determination of the DSPI from a realized $\overline{\kappa}$ score gives

$$\theta^*_{\text{opt}} = \begin{cases} 1 & \text{if } \{N < P\}, \\ 0 & \text{if } \{P < N\}, \\ \Theta^* & \text{if } \{N = P\}. \end{cases}$$

By filling this θ^*_{opt}, the following κ_α formulation is obtained

$$\kappa_\alpha = \frac{\alpha \cdot (1 - 2 \cdot \rho) \cdot 2 \cdot P \cdot N}{\alpha \cdot (\rho \cdot (N - P)^2 - (\min\{P, N\})^2 + P \cdot N) + M \cdot \min\{P, N\}}. \tag{5.23}$$

If we would look at the boundaries of this function, it can be observed that $\kappa_0 = 0$, which equals the DD baseline, and $\kappa_1 = \frac{2 \cdot P \cdot N \cdot (1-2 \cdot \rho)}{\rho \cdot (N-P)^2 + 2 \cdot P \cdot N}$ indicates the highest performance value.

Derivatives: The DS requires $\frac{\partial \mu_\alpha}{\partial \alpha} > 0$ for $\forall \alpha \in [0, 1]$ to have a strictly increasing performance score in α. The derivative of κ_α (using Equation (5.23)) with respect to α is given by

$$\frac{\partial \kappa_\alpha}{\partial \alpha} = \frac{(1 - 2 \cdot \rho) \cdot 2 \cdot P \cdot N \cdot M \cdot \min\{P, N\}}{(\alpha \cdot (\rho \cdot (N-P)^2 + P \cdot N - (\min\{P, N\})^2) + M \cdot \min\{P, N\})^2}.$$

Thus, to satisfy the mentioned requirement, it should hold that $\rho < \frac{1}{2}$. Let us look at the second derivative and whether the function is concave up/down. It is given by

$$\frac{\partial^2 \kappa_\alpha}{\partial \alpha^2} = \frac{-2 \cdot (1 - 2 \cdot \rho) \cdot 2 \cdot P \cdot N \cdot M \cdot \min\{P, N\} \cdot V}{(\alpha \cdot (\rho \cdot (N-P)^2 + P \cdot N - (\min\{P, N\})^2) + M \cdot \min\{P, N\})^3},$$

where $V = \rho \cdot (N-P)^2 + P \cdot N - (\min\{P, N\})^2$. This is always negative, so the function is concave down.

Upper Bound: Let us denote Ω_κ as the expected κ score of the 'optimal' model, and its value is known. To satisfy the desirable property that this value equals μ_1, ρ should be

$$\rho = 2 \cdot P \cdot N \cdot \frac{1 - \Omega_\kappa}{\Omega_\kappa \cdot (N-P)^2 + 4 \cdot P \cdot N}. \tag{5.24}$$

Determining the DSPI: Let us show the optimal θ^*_{opt} for the κ_α. Solving Equation (5.22) for α gives

$$\alpha = \frac{\theta^*_{\text{opt}} \cdot (N-P) \cdot M \cdot \overline{\kappa} + P \cdot M \cdot \overline{\kappa}}{\theta^*_{\text{opt}} \cdot M \cdot \overline{\kappa} \cdot (N-P) + \overline{\kappa} \cdot (P-N) \cdot (\rho \cdot (N-P) + P) + 2 \cdot P \cdot N \cdot (1 - 2 \cdot \rho)}, \tag{5.25}$$

with $\overline{\kappa}$ being the realized Kappa score. Note that α quantifies how much the model has learned. To see whether α is decreasing or increasing in θ^*_{opt}, we determine the derivative of the function $f : [0, 1] \to [0, 1]$, given by

$$f(\theta) = \frac{\theta \cdot \overline{\kappa} \cdot M \cdot (N-P) + \overline{\kappa} \cdot M \cdot P}{\theta \cdot M \cdot \overline{\kappa} \cdot (N-P) + \overline{\kappa} \cdot (P-N) \cdot (\rho \cdot (N-P) + P) + 2 \cdot P \cdot N \cdot (1 - 2 \cdot \rho)}.$$

This function is the continuous version of Equation (5.22), note that $\alpha = f(\theta^*_{\text{opt}})$. Taking the derivative gives us

$$\frac{\partial f(\theta)}{\partial \theta} = \frac{\overline{\kappa} \cdot M \cdot (N-P) \cdot (-\rho \cdot \overline{\kappa} \cdot (P-N)^2 + 2 \cdot N \cdot P \cdot (1 - \overline{\kappa}) - 4 \cdot P \cdot N \cdot \rho)}{(\theta \cdot M \cdot \overline{\kappa} \cdot (N-P) + \overline{\kappa} \cdot (P-N) \cdot (\rho \cdot (N-P) + P) + G)^2},$$

where $G = 2 \cdot P \cdot N \cdot (1 - 2 \cdot \rho)$. The last term in the numerator is always non-negative due to Equation (5.24). It can be observed that the denominator is always

non-negative, but the numerator is either non-negative or non-positive depending on whether $N < P$ or $P < N$. We can see that the sign depends on whether P or N is larger. If $N > P$, then the function is increasing, and we select the lowest value, $\theta^*_{\text{opt}} = 0$, while if P is larger, we select $\theta^*_{\text{opt}} = 1$. If $P = N$, θ^*_{opt} is left out of the equation and can be anything in Θ^*. Filling in this strategy in Equation (5.25) gives us

$$\alpha = \frac{M \cdot \min\{P,N\} \cdot \overline{\kappa}}{\overline{\kappa} \cdot ((\min\{P,N\})^2 - N \cdot P - \rho \cdot (P-N)^2) + 2 \cdot P \cdot N \cdot (1 - 2 \cdot \rho)}.$$

5

Scaling the DSPI: Equation (5.23) can be analyzed in its behavior by scaling the scores with a min-max scaler between μ_0 and μ_1 with the PP ($\rho = 0$). This gives us

$$\frac{\overline{\kappa}|\rho=0 - \mu_0}{\mu_1|\rho=0 - \mu_0} = \frac{2 \cdot P \cdot N \cdot \alpha}{\alpha \cdot (P \cdot N - (\min\{P,N\})^2) + M \cdot \min\{P,N\}}.$$

5.8.10 F_β score

The F_β *score* (F^β_α) under the DS is defined as follows:

$$\begin{aligned} F^\beta_\alpha &= \frac{1+\beta^2}{\frac{1}{\text{PPV}_\alpha} + \frac{\beta^2}{\text{TPR}_\alpha}} \\ &= \frac{(1+\beta^2) \cdot (\alpha \cdot P \cdot (1 - \rho - \theta^*_{\text{opt}}) + P \cdot \theta^*_{\text{opt}})}{\alpha \cdot (\rho \cdot (N-P) + P - M \cdot \theta^*_{\text{opt}}) + M \cdot \theta^*_{\text{opt}} + P \cdot \beta^2}, \end{aligned} \tag{5.26}$$

by using Equations (5.7) and (5.9). The optimal DD strategy for F^β_α is $\theta^*_{\text{opt}} = 1$. Hence, Equation (5.26) simplifies to

$$F^\beta_\alpha = P \cdot (1+\beta^2) \cdot \frac{1 - \alpha \cdot \rho}{\alpha \cdot (\rho \cdot (N-P) - N) + M + P \cdot \beta^2}. \tag{5.27}$$

From this definition, it can be derived that $F^\beta_0 = \frac{P \cdot (1+\beta^2)}{M + P \cdot \beta^2}$, which equals the DD baseline, and $F^\beta_1 = \frac{P \cdot (1+\beta^2) \cdot (1-\rho)}{\rho \cdot (N-P) + P \cdot (1+\beta^2)}$, indicating the highest performance value.

Derivatives: The DS requires $\dfrac{\partial \mu_\alpha}{\partial \alpha} > 0$ for $\forall \alpha \in [0,1]$ to have a strictly increasing performance score in α. The derivative of F^β_α with respect to α is given by

$$\frac{\partial F^\beta_\alpha}{\partial \alpha} = P \cdot (1+\beta^2) \cdot \frac{N \cdot (1 - 2 \cdot \rho) - \rho \cdot P \cdot \beta^2}{(\alpha \cdot (\rho \cdot (N-P) - N) + M + P \cdot \beta^2)^2}.$$

As the denominator and the first two terms are always positive, F_α^β is strictly increasing in α if

$$\rho < \frac{N}{2 \cdot N + P \cdot \beta^2}. \tag{5.28}$$

The second derivative of F_α^β in α is given by

$$\frac{\partial^2 F_\alpha^\beta}{\partial \alpha^2} = P \cdot (1+\beta^2) \cdot \frac{2 \cdot (N \cdot (1-\rho) + P \cdot \rho) \cdot (N \cdot (1-2 \cdot \rho) - \rho \cdot P \cdot \beta^2)}{(\alpha \cdot (\rho \cdot (N-P) - N) + M + P \cdot \beta^2)^3}.$$

This function is always non-negative due to Equation (5.28), showing that the function is concave up.

Upper Bound: Let us denote Ω_{F^β} as the expected F^β score of the 'optimal' model, and its value is known. To satisfy the desirable property that this value equals μ_1, ρ should be

$$\rho = \frac{(1+\beta^2) \cdot P \cdot (1 - \Omega_{F^\beta})}{\Omega_{F^\beta} \cdot (N-P) + (1+\beta^2) \cdot P}.$$

Determining the DSPI: Suppose we have a realized $\overline{F^\beta}$ at our disposal. The DSPI can be derived by taking the inverse function of Equation (5.27), as we already excluded θ^*_{opt} from the equation. The DSPI is thus determined by

$$\alpha = \frac{P \cdot (1+\beta^2)(1-\overline{F^\beta}) - \overline{F^\beta} \cdot N}{\overline{F^\beta} \cdot (\rho \cdot (N-P) - N) + (1+\beta^2) \cdot P \cdot \rho}.$$

Scaling the DSPI: Equation (5.27) can be analyzed in its behavior by scaling the scores with a min-max scaler between μ_0 and μ_1 with the PP ($\rho = 0$). This gives us

$$\frac{F_\alpha^\beta | \rho = 0 - \mu_0}{\mu_1 | \rho = 0 - \mu_0} = \frac{\frac{(1+\beta^2) \cdot P}{\alpha \cdot -N + M + P \cdot \beta^2} - \frac{(1+\beta^2) \cdot P}{\beta^2 \cdot P + M}}{1 - \frac{(1+\beta^2) \cdot P}{\beta^2 \cdot P + M}} = \alpha \cdot \frac{(1+\beta^2) \cdot P}{M - \alpha \cdot N + P \cdot \beta^2}.$$

5.8.11 Fowlkes-Mallows Index

The *Fowlkes-Mallows index* (FM_α) under the DS is defined as follows:

$$\begin{aligned} \text{FM}_\alpha &:= \sqrt{\text{TPR}_\alpha \cdot \text{PPV}_\alpha} = \frac{\text{TP}_\alpha}{\sqrt{P \cdot (\text{TP}_\alpha + \text{FP}_\alpha)}} \\ &= \sqrt{P} \cdot \frac{\alpha \cdot (1 - \rho - \theta^*_{\text{opt}}) + \theta^*_{\text{opt}}}{\sqrt{\alpha \cdot (\rho \cdot (N-P) + P - M \cdot \theta^*_{\text{opt}}) + M \cdot \theta^*_{\text{opt}}}}, \end{aligned} \tag{5.29}$$

by using Equations (5.1), (5.5), (5.7) and (5.9). The optimal DD strategy for FM_α is $\theta^*_{\text{opt}} = 1$. Hence, Equation (5.29) simplifies to

$$\text{FM}_\alpha \coloneqq \sqrt{P} \cdot \frac{1 - \alpha \cdot \rho}{\sqrt{\alpha \cdot (\rho \cdot (N - P) - N) + M}}. \tag{5.30}$$

From the definition, it can be derived that $\text{FM}_0 = \sqrt{\frac{P}{M}}$, which equals the DD baseline, and $\text{FM}_1 = \frac{\sqrt{P} \cdot (1-\rho)}{\sqrt{\rho \cdot (N-P)+P}}$, indicating the highest performance value.

Derivatives: The DS requires $\frac{\partial \mu_\alpha}{\partial \alpha} > 0$ for $\forall \alpha \in [0, 1]$ to have a strictly increasing performance score in α. The derivative of FM_α with respect to α is given by

$$\frac{\partial \text{FM}_\alpha}{\partial \alpha} = \sqrt{P} \cdot \frac{\alpha \cdot \rho^2 \cdot P + \alpha \cdot N \cdot \rho \cdot (1 - \rho) + N - \rho \cdot (3 \cdot N + P)}{2 \cdot (M - \alpha \cdot (N \cdot (1 - \rho) + P \cdot \rho))^{\frac{3}{2}}}.$$

It can be observed that the denominator is always positive as $N \cdot (1-\rho) + P \cdot \rho \leq N + P = M$ and thus $M - \alpha \cdot (N \cdot (1 - \rho) + P \cdot \rho) = \hat{P} \geq 0$. To satisfy the requirement, it should hold that

$$\rho < \frac{N}{3N + P}.$$

Only then is the function non-decreasing $\forall \alpha \in [0, 1]$. If we now look at the second derivative, which is given by

$$\frac{\partial^2 \text{FM}_\alpha}{\partial \alpha^2} = \sqrt{P} \cdot \frac{L + \rho \cdot (-7 \cdot N - P) + 3 \cdot N)}{4 \cdot (M - \alpha \cdot (N \cdot (1 - \rho) + P \cdot \rho))^{\frac{5}{2}}},$$

where $L = (N \cdot (1 - \rho) + \rho \cdot P) \cdot (\alpha \cdot \rho \cdot (N \cdot (1 - \rho) + P \cdot \rho)$. Now if $\rho < \frac{3 \cdot N}{7 \cdot N + P}$, the derivative is always positive, and thus the function is concave up. As we have the restriction on the first derivative $\rho < \frac{N}{3N+P}$, which is more strict, the function is always concave up.

Upper Bound: Let us denote Ω_{FM} as the expected FM score of the 'optimal' model, and its value is known. To satisfy the desirable property that this value equals μ_1, ρ should be

$$\Omega_{\text{FM}} = \frac{\sqrt{P} \cdot (1 - \rho)}{\sqrt{P \cdot (1 - \rho) + \rho \cdot N}}.$$

Solving this inequality can be solved by solving the following inequality, using $g = \Omega_{\text{FM}}$:

$$\rho^2 + \rho \cdot (\frac{P - N}{P} \cdot g^2 - 2) + (1 - g^2) \geq 0.$$

The discriminant of this problem is given by

$$D = g^2 \cdot ((\frac{P-N}{P})^2 \cdot g^2 + 4 \cdot \frac{N}{P}) > 0.$$

This is always positive, so there are always two solutions. The quadratic solution gives us

$$\rho = 1 + \frac{-\frac{P-N}{P} \cdot g^2 \pm g \cdot \sqrt{(\frac{P-N}{P})^2 \cdot g^2 + 4 \cdot \frac{N}{P}}}{2}.$$

Let us now look at the first solution (+):

$$1 + \frac{-\frac{P-N}{P} \cdot g^2 + g \cdot \sqrt{(\frac{P-N}{P})^2 \cdot g^2 + 4 \cdot \frac{N}{P}}}{2} > 1 + \frac{-\frac{P-N}{P} \cdot g^2 + g \cdot \sqrt{(\frac{P-N}{P})^2 \cdot g^2}}{2} = 1.$$

This implies that this solution gives some $\rho > 1$, but this is out of the range of ρ. For the other solution (−), it holds that

$$1 + g \cdot (\frac{\frac{N-P}{P} \cdot g - \sqrt{(\frac{N-P}{P})^2 \cdot g^2 + 4 \cdot \frac{N}{P}}}{2}) \leq 1.$$

If we show that the second derivative is positive, we know that only the second solution is valid. We can directly see that the second derivative of this function is always positive as all other terms are also positive and, therefore, the other solution results in $\rho \leq 1$. Still, we need to show it is above 0, and if we plug in $\rho = 0$, we get $(1 - g^2) \geq 0$, which always holds, so the solution must be between $0 \leq \rho \leq 1$. This combined implies that we have the following inequality:

$$\rho = 1 - \frac{\frac{P-N}{P} \cdot (\Omega_{\text{FM}})^2 + (\Omega_{\text{FM}_\alpha}) \cdot \sqrt{\left(\frac{P-N}{P}\right)^2 \cdot (\Omega_{\text{FM}_\alpha})^2 + 4 \cdot \frac{N}{P}}}{2}.$$

Determining the DSPI: Let us show the optimal θ^*_{opt} for the FM_α. Solving Equation (5.30) for α gives

$$\alpha^2 \cdot \rho^2 \cdot P - \alpha \cdot \Big(2 \cdot \rho \cdot P + \overline{\text{FM}}^2 \cdot (\rho \cdot (N - P) - N)\Big) = M \cdot \overline{\text{FM}}^2 - P,$$

with $\overline{\text{FM}}$ the realized value of FM_α. If $\rho = 0$, we have

$$\alpha = \frac{M}{N} - \frac{P}{(\overline{\text{FM}})^2 \cdot N}.$$

Let us use again $g = \overline{\text{FM}}$. Solving the equation in α yields

$$\alpha = \frac{1}{\rho} + \frac{g^2 \cdot (\rho \cdot (N - P) - N) \pm \sqrt{D}}{2 \cdot \rho^2 \cdot P},$$

with

$$D = g^2 \cdot \left(g^2 \cdot (\rho \cdot (P - N) + N)^2 + 8 \cdot \rho^2 \cdot P^2 + N \cdot 4 \cdot \rho \cdot P\right) \geq 0.$$

Giving two solutions. It is an open question which one of the two solutions results in $\alpha \in [0, 1]$.

Scaling the DSPI: Equation (5.30) can be analyzed in its behavior by scaling the scores with a min-max scaler between μ_0 and μ_1 with the PP ($\rho = 0$). This gives us

$$\frac{\overline{\text{FM}}|\rho = 0 - \mu_0}{\mu_1|\rho = 0 - \mu_0} = \frac{\frac{\sqrt{P}}{\sqrt{M - \alpha \cdot N}} - \sqrt{\frac{P}{M}}}{1 - \sqrt{\frac{P}{M}}}.$$

We could not further simplify this function.

5.8.12 G-mean 2

The *G-mean 2* ($\text{G}_\alpha^{(2)}$) under the DS is defined as follows:

$$\begin{aligned} \text{G}_\alpha^{(2)} &:= \sqrt{\text{TPR}_\alpha \cdot \text{TNR}_\alpha} = \sqrt{\frac{\text{TP}_\alpha}{P} \cdot \frac{\text{TN}_\alpha}{N}} \\ &= \sqrt{\left(\alpha \cdot (1 - \rho - \theta^*_{\text{opt}}) + \theta^*_{\text{opt}}\right) \cdot \left(\alpha \cdot (\theta^*_{\text{opt}} - \rho) + (1 - \theta^*_{\text{opt}})\right)}, \end{aligned} \tag{5.31}$$

by using Equations (5.1), (5.2), (5.7) and (5.8). The optimal DD strategy for $\text{G}_\alpha^{(2)}$ is (until now) unknown, but we know that the upper limit of the $\text{G}_\alpha^{(2)}$ is $\text{G}_1^{(2)} = \sqrt{(1 - \rho) \cdot (1 - \rho)} = 1 - \rho$ and $\text{G}_0^{(2)}$ gives $\sqrt{(\theta^*_{\text{opt}}) \cdot (1 - \theta^*_{\text{opt}})}$. Experimentation shows that this is not equal to the DD baseline.

Derivatives: The DS requires $\frac{\partial \mu_\alpha}{\partial \alpha} > 0$ for $\forall \alpha \in [0, 1]$ to have a strictly increasing performance score in α. The derivative with respect to α of the $\text{G}_\alpha^{(2)}$ under the DS is given by

$$\frac{\partial \text{G}_\alpha^{(2)}}{\partial \alpha} = \frac{2 \cdot \alpha \cdot \rho^2 - (2 \cdot \alpha + 1) \cdot \rho + 2 \cdot (1 - \alpha) \cdot (\theta^*_{\text{opt}})^2 + 2 \cdot (\alpha - 1) \cdot \theta^*_{\text{opt}} + 1}{2 \cdot \sqrt{\left(\alpha \cdot (1 - \rho - \theta^*_{\text{opt}}) + \theta^*_{\text{opt}}\right) \cdot \left(\alpha \cdot (\theta^*_{\text{opt}} - \rho) + (1 - \theta^*_{\text{opt}})\right)}}.$$

It can be observed that the denominator is always non-negative or not-defined. Therefore, whether or not the derivative is always positive depends on the numerator. ρ should satisfy the following inequality:

$$2 \cdot \alpha \cdot \rho^2 - (2 \cdot \alpha + 1) \cdot \rho + 2 \cdot (1 - \alpha) \cdot (\theta^*_{\text{opt}})^2 + 2 \cdot (\alpha - 1) \cdot \theta^*_{\text{opt}} + 1 > 0, \forall \alpha \in [0, 1].$$

We can write this in terms of α:

$$\alpha \cdot 2 \cdot (\rho \cdot (\rho - 1) + \theta^*_{\text{opt}} \cdot (1 - \theta^*_{\text{opt}})) > -2 \cdot (\theta^*_{\text{opt}})^2 + 2 \cdot \theta^*_{\text{opt}} - 1 + \rho, \forall \alpha \in [0, 1].$$

We have two situations here. Suppose $\rho \cdot (\rho - 1) + \theta^*_{\text{opt}} \cdot (1 - \theta^*_{\text{opt}}) \geq 0$, then the function decreases in α and we should therefore select the lowest $\alpha = 0$ that should be satisfied. This results in

$$0 \cdot 2 \cdot (\rho \cdot (\rho - 1) + \theta^*_{\text{opt}} \cdot (1 - \theta^*_{\text{opt}})) > -2 \cdot (\theta^*_{\text{opt}})^2 + 2 \cdot \theta^*_{\text{opt}} - 1 + \rho, \forall \alpha \in [0, 1].$$

Leading to $\rho < 1 + 2 \cdot \theta^*_{\text{opt}} \cdot (\theta^*_{\text{opt}} - 1)$, but, we also assumed that $\theta^*_{\text{opt}} \cdot (\theta^*_{\text{opt}} - 1) \leq \rho \cdot (\rho - 1)$. So this inequality leads to

$$\rho < 1 + 2 \cdot \theta^*_{\text{opt}} \cdot (\theta^*_{\text{opt}} - 1) \leq 1 - 2 \cdot \rho \cdot (1 - \rho) = 1 + 2 \cdot \rho^2 - 2 \cdot \rho.$$

This is only true when $\rho < \frac{1}{2}$. Now, let us look at the other situation. Suppose $\rho \cdot (\rho - 1) + \theta^*_{\text{opt}} \cdot (1 - \theta^*_{\text{opt}}) \leq 0$. This implies we have to select $\alpha = 1$. Substituting this $alpha$ gives us

$$2 \cdot (\rho \cdot (\rho - 1) + \theta^*_{\text{opt}} \cdot (1 - \theta^*_{\text{opt}})) > 2 \cdot \theta^*_{\text{opt}} \cdot (1 - \theta^*_{\text{opt}}) - 1 + \rho 2 \cdot \rho^2 - 2 \cdot \rho > -1 + \rho.$$

Giving the same inequality to solve, also in this situation $\rho < \frac{1}{2}$. So, to satisfy the assumption, we need to assume $\rho < \frac{1}{2}$. Now, let us look at the second derivatve, which is given by

$$\frac{\partial^2 \text{G}^{(2)}_\alpha}{\partial \alpha^2} = -\frac{(\rho - 1)^2 \cdot (1 - 2 \cdot \theta^*_{\text{opt}})^2}{4 \cdot \left(\left(\alpha \cdot (1 - \rho - \theta^*_{\text{opt}}) + \theta^*_{\text{opt}}\right) \cdot \left(\alpha \cdot (\theta^*_{\text{opt}} - \rho) + (1 - \theta^*_{\text{opt}})\right)\right)^{\frac{3}{2}}}.$$

This is always non-positive, meaning the function will be concave down except when θ^*_{opt} is always 0.5.

Upper Bound: Let us denote $\Omega_{\text{G}^{(2)}}$ as the expected G-mean 2 score of the 'optimal' model, and its value is known. To satisfy the desirable property that this value equals μ_1, ρ should be

$$\rho = 1 - \Omega_{\text{G}^{(2)}}.$$

Determining the DSPI: Suppose we have a realized $\overline{\text{G}}^{(2)}$ at our disposal. The DSPI is derived by inverting Equation (5.31) in α. It is shown that the $\text{G}^{(2)}_\alpha$ is not linear in $\text{TP}^{\text{DD}}_\theta$. The DD baseline maximizers are unknown. Solving Equation (5.31) for α gives

$$\alpha^2 \cdot (1 - \rho - \theta^*_{\text{opt}}) \cdot (\theta^*_{\text{opt}} - \rho) + \alpha \cdot (1 - \rho - 2 \cdot \theta^*_{\text{opt}} + 2 \cdot (\theta^*_{\text{opt}})^2) + \theta^*_{\text{opt}} \cdot (1 - \theta^*_{\text{opt}}) - (\overline{\text{G}}^{(2)})^2 = 0.$$

If $(1-\rho-\theta^*_{\text{opt}})\cdot(\theta^*_{\text{opt}}-\rho)=0$, we have

$$\alpha=\frac{(\overline{\text{G}}^{(2)})^2-\theta^*_{\text{opt}}\cdot(1-\theta^*_{\text{opt}})}{(1-\rho-2\cdot\theta^*_{\text{opt}}+2\cdot(\theta^*_{\text{opt}})^2)}.$$

In the other situations, we have a quadratic problem, and solving this gives

$$\alpha=\frac{(\rho-1)+2\cdot\theta^*_{\text{opt}}\cdot(1-\theta^*_{\text{opt}})\pm\sqrt{D}}{2\cdot\rho\cdot(\rho-1)+2\cdot\theta^*_{\text{opt}}\cdot(1-\theta^*_{\text{opt}})},$$

5

with $D=(1-\rho-2\cdot\theta^*_{\text{opt}}+2\cdot(\theta^*_{\text{opt}})^2)^2-4\cdot(1-\rho-\theta^*_{\text{opt}})\cdot(\theta^*_{\text{opt}}-\rho)\cdot(\theta^*_{\text{opt}}\cdot(1-\theta^*_{\text{opt}})-(\overline{\text{G}}^{(2)})^2)$. We already said that $\rho<0.5$ to fulfill the requirement that the DSPI is strictly increasing in α. This means that only one answer will be valid as the summation will give a $\alpha\in[0,1]$.

Scaling the DSPI: Equation (5.31) can be analyzed in its behavior by scaling the scores with a min-max scaler between μ_0 and μ_1 with the PP ($\rho=0$). This gives us

$$\frac{\overline{\text{G}}^{(2)}|\rho=0-\mu_0}{\mu_1|\rho=0-\mu_0}=\frac{K-\sqrt{(\theta^*_{\text{opt}})\cdot(1-\theta^*_{\text{opt}})}}{1-\sqrt{(\theta^*_{\text{opt}})\cdot(1-\theta^*_{\text{opt}})}},$$

where $K=\sqrt{\left(\alpha\cdot(1-\theta^*_{\text{opt}})+\theta^*_{\text{opt}}\right)\cdot\left(\alpha\cdot\theta^*_{\text{opt}}+(1-\theta^*_{\text{opt}})\right)}$. This function is not easily simplified.

5.8.13 Markedness

The *markedness* (MK_α) under the DS is defined as follows:

$$\begin{aligned}\text{MK}_\alpha\coloneqq\text{PPV}_\alpha+\text{NPV}_\alpha-1=&\frac{\alpha\cdot P\cdot(1-\rho-\theta^*_{\text{opt}})+P\cdot\theta^*_{\text{opt}}}{\alpha\cdot(\rho\cdot(N-P)+P-M\cdot\theta^*_{\text{opt}})+M\cdot\theta^*_{\text{opt}}}\\&+\frac{\alpha\cdot N\cdot(\theta^*_{\text{opt}}-\rho)+N\cdot(1-\theta^*_{\text{opt}})}{\alpha\cdot(\rho\cdot(P-N)+M\cdot\theta^*_{\text{opt}}-P)+M\cdot(1-\theta^*_{\text{opt}})}-1,\end{aligned}\tag{5.32}$$

by using Equations (5.9) and (5.13). We can rewrite this definition as follows:

$$\text{MK}_\alpha\coloneqq\alpha\cdot(1-2\cdot\rho)\cdot\frac{P\cdot N}{\hat{P}\cdot\hat{N}},$$

where $\hat{P}\cdot\hat{N}=-\alpha^2\cdot(\rho\cdot(N-P)+P-M\cdot\theta^*_{\text{opt}})^2+\alpha\cdot M\cdot(\rho\cdot(N-P)+P-M\cdot\theta^*_{\text{opt}})\cdot(1-2\cdot\theta^*_{\text{opt}})+M^2\cdot\theta^*_{\text{opt}}\cdot(1-\theta^*_{\text{opt}})$. The optimal DD strategy for PPV_α is $\theta^*_{\text{opt}}\in\Theta^*\setminus\{0,1\}$. Hence, there is not a unique θ^*_{opt} resulting in the

DD baseline. From this definition, it can be derived that $\text{MK}_0 = 0$, which equals the DD baseline, and $\text{MK}_1 = N \cdot P \cdot \frac{1-2\cdot\rho}{-\rho^2\cdot(P-N)^2+\rho\cdot(P-N)^2+N\cdot P}$, indicating the highest performance value.

Derivations To respect the assumption that $\frac{\partial \mu_\alpha}{\partial \alpha} > 0$ for $\forall \alpha \in [0,1]$, we should check for which ρ this is the case. Its derivative is given by

$$\frac{\partial \text{MK}_\alpha}{\partial \alpha} = (1-2\cdot\rho)\cdot P \cdot N \cdot \frac{\alpha^2\cdot(\rho\cdot(N-P)+P-M\cdot\theta^*_{\text{opt}})^2+G}{(\hat{P}\cdot\hat{N})^2},$$

where $G = M^2\cdot\theta^*_{\text{opt}}\cdot(1-\theta^*_{\text{opt}})$. We can see that the derivative is non-decreasing in α if $\rho < 0.5$. Let us now look at the second derivative, which is given by

$$\frac{(\hat{P}\cdot\hat{N})^3}{2\cdot(1-2\cdot\rho)\cdot P\cdot N}\cdot\frac{\partial^2\text{MK}_\alpha}{\partial\alpha^2} = \alpha^3\cdot(\rho\cdot(N-P)+P-M\cdot\theta^*_{\text{opt}})^4$$
$$+M^2\cdot\theta^*_{\text{opt}}\cdot(1-\theta^*_{\text{opt}})\cdot(\rho\cdot(N-P)+P-M\cdot\theta^*_{\text{opt}})\cdot(2\cdot\hat{P}-M).$$

It can be observed that a simple derivation is not easily obtained.

Upper Bound: Let us denote Ω_{MK} as the expected MK score of the 'optimal' model, and its value is known. To satisfy the desirable property that this value equals μ_1, ρ should be

$$\Omega_{\text{MK}} = N\cdot P\cdot\frac{1-2\cdot\rho}{-\rho^2\cdot(P-N)^2+\rho\cdot(P-N)^2+N\cdot P}.$$

Turning this in ρ gives the following equality which should hold for ρ:

$$0 = \rho^2\cdot\Omega_{\text{MK}}\cdot(P-N)^2+N\cdot P\cdot(1-\Omega_{\text{MK}})$$
$$-\rho\cdot(\Omega_{\text{MK}}\cdot(P-N)^2+2\cdot N\cdot P).$$

Solving the equality (=) gives

$$\rho = \frac{1}{2}+\frac{2\cdot N\cdot P\pm\sqrt{D}}{2\cdot g\cdot(P-N)^2},$$

with $g = \Omega_{\text{MK}}$ and

$$D = g^2\cdot(P-N)^4+4\cdot N\cdot P\cdot g^2\cdot(P-N)^2+4\cdot N^2\cdot P^2.$$

The solution corresponding to the $+$ sign exceeds 1 and is therefore invalid:

$$\frac{1}{2}+\frac{2\cdot N\cdot P+\sqrt{D}}{2\cdot g\cdot(P-N)^2} > 1.$$

Thus, the valid solution is given by the $-$ sign:

$$\frac{1}{2} + \frac{2 \cdot N \cdot P - \sqrt{D}}{2 \cdot g \cdot (P - N)^2}.$$

An open question is whether this results in a $\rho \in [0, 1]$.

Determining the DSPI: Suppose we have a realized $\overline{\text{MK}}$ at our disposal. Trying to write $\overline{\text{MK}}$ in terms of α results in the following equation

$$\begin{aligned} &\alpha^2 \cdot (\rho \cdot (N - P) + P - M \cdot \theta^*_{\text{opt}})^2 \cdot \overline{\text{MK}} \\ -\alpha \cdot (M \cdot (\rho \cdot (N - P) + P - M \cdot \theta^*_{\text{opt}}) \cdot (1 - 2 \cdot \theta^*_{\text{opt}}) \cdot \overline{\text{MK}} &- (1 - 2 \cdot \rho) \cdot P \cdot N) \\ &- M^2 \cdot \theta^*_{\text{opt}} \cdot (1 - \theta^*_{\text{opt}}) \cdot \overline{\text{MK}} = 0. \end{aligned}$$

If $\rho \cdot (N - P) + P - M \cdot \theta^*_{\text{opt}} = 0$, we have

$$\alpha = \frac{M^2 \cdot \theta^*_{\text{opt}} \cdot (1 - \theta^*_{\text{opt}}) \cdot \overline{\text{MK}}}{(1 - 2 \cdot \rho) \cdot P \cdot N}.$$

If this is not the case, we have a quadratic problem with solutions depending on the discriminant of the quadratic equation:

$$\begin{aligned} D = (M \cdot (\rho \cdot (N - P) + P - M \cdot \theta^*_{\text{opt}}) \cdot (1 - 2 \cdot \theta^*_{\text{opt}}) \cdot \overline{\text{MK}} - (1 - 2 \cdot \rho) \cdot P \cdot N)^2 \\ + 4 \cdot (\rho \cdot (N - P) + P - M \cdot \theta^*_{\text{opt}})^2 \cdot (\overline{\text{MK}})^2 \cdot M^2 \cdot \theta^*_{\text{opt}} \cdot (1 - \theta^*_{\text{opt}}). \end{aligned}$$

As the second term is positive, we know that $\frac{-b - \sqrt{D}}{2 \cdot a} < 0$ and we can observe that the $D > 0$ by $\rho < 0.5$, so we have two solutions. The only possible solution is

$$\alpha = \frac{M \cdot (\rho \cdot (N - P) + P - M \cdot \theta^*_{\text{opt}}) \cdot (1 - 2 \cdot \theta^*_{\text{opt}}) \cdot \overline{\text{MK}} + G}{2 \cdot (\rho \cdot (N - P) + P - M \cdot \theta^*_{\text{opt}})^2 \cdot \overline{\text{MK}}},$$

where $G = \sqrt{D} - (1 - 2 \cdot \rho) \cdot P \cdot N$.

Scaling the DSPI: Equation (5.32) can be analyzed in its behavior by scaling the scores with a min-max scaler between μ_0 and μ_1 with the PP ($\rho = 0$). This gives us

$$\frac{\overline{\text{MK}}|\rho = 0 - \mu_0}{\mu_1|\rho = 0 - \mu_0} = \alpha \cdot \frac{P \cdot N}{\hat{P} \cdot \hat{N}}.$$

5.8.14 Matthews' Correlation Coefficient

The *Matthews' correlation coefficient* (MCC_α) under the DS is given by

$$\text{MCC}_\alpha := \frac{\text{TP}_\alpha \cdot \text{TN}_\alpha - \text{FN}_\alpha \cdot \text{FP}_\alpha}{\sqrt{\hat{P} \cdot P \cdot \hat{N} \cdot N}} = \alpha \cdot (1 - 2 \cdot \rho) \cdot \sqrt{\frac{P \cdot N}{\hat{P} \cdot \hat{N}}}, \tag{5.33}$$

where $\hat{P} \cdot \hat{N} = -\alpha^2 \cdot (\rho \cdot (N-P) + P - M \cdot \theta^*_{\text{opt}})^2 + \alpha \cdot M \cdot (\rho \cdot (N-P) + P - M \cdot \theta^*_{\text{opt}}) \cdot (1 - 2 \cdot \theta^*_{\text{opt}}) + M^2 \cdot \theta^*_{\text{opt}} \cdot (1 - \theta^*_{\text{opt}})$. The optimal DD strategy is given by $\Theta^* \setminus \{0, 1\}$, and there is not a unique θ^*_{opt} resulting in the DD baseline. Looking at the boundaries of the DS, it can be observed that $\text{MCC}_0 = 0$, which equals the DD baseline, and $\text{MCC}_1 = \frac{(1-2\cdot\rho)\cdot\sqrt{P\cdot N}}{\sqrt{(\rho\cdot(N-P)+P)\cdot(\rho\cdot(P-N)+N)}}$ indicates the highest performance value.

Derivatives: To respect the requirement that $\frac{\partial \mu_\alpha}{\partial \alpha} > 0$ for $\forall \alpha \in [0, 1]$, we should check for which ρ this is the case. Its derivative is given by

$$\frac{\partial \text{MCC}_\alpha}{\partial \alpha} = (1 - 2 \cdot \rho) \cdot M \cdot \sqrt{P \cdot N} \cdot \frac{\hat{P} \cdot (1 - \theta^*_{\text{opt}}) + \hat{N} \cdot \theta^*_{\text{opt}}}{(\hat{P} \cdot \hat{N})^{\frac{3}{2}}}.$$

Showing that we require $\rho < \frac{1}{2}$ to have μ_α strictly increasing. Let us now look at the second derivatve, which is given by

$$\frac{\partial^2 \text{MCC}_\alpha}{\partial \alpha^2} = (1 - 2 \cdot \rho) \cdot M \cdot \sqrt{P \cdot N} \cdot (\rho \cdot (N - P) + P - M \cdot \theta^*_{\text{opt}}) \cdot E,$$

where

$$E = \frac{(1 - \theta^*_{\text{opt}}) \cdot (4 \cdot \hat{P} - M)}{2 \cdot \hat{P}^{\frac{3}{2}} \cdot \hat{N}^{\frac{5}{2}}} + \frac{\theta^*_{\text{opt}} \cdot (M - 4 \cdot \hat{N})}{2 \cdot \hat{N}^{\frac{3}{2}} \cdot \hat{P}^{\frac{5}{2}}}.$$

Whether the metric is concave up/down on α is dependent on the specified parameters.

Upper Bound: Let us denote Ω_{MCC} as the expected MCC score of the 'optimal' model, and its value is known. To satisfy the desirable property that this value equals μ_1, ρ should be

$$\Omega_{\text{MCC}} = \frac{\sqrt{P \cdot N} \cdot (1 - 2 \cdot \rho)}{\sqrt{(\rho \cdot (N - P) + P) \cdot (\rho \cdot (P - N) + N)}}.$$

Let us say $g = \Omega_{\text{MCC}}$. Then the equation to solve is

$$\rho^2 - \rho + \frac{P \cdot N \cdot (1 - g^2)}{g^2 \cdot (N - P)^2 + 4 \cdot P \cdot N} = 0.$$

This is a quadratic problem, and the discriminant of this quadratic equation is

$$D = 1 - 4 \cdot \frac{P \cdot N \cdot (1 - g^2)}{g^2 \cdot (N - P)^2 + 4 \cdot P \cdot N} = \frac{g^2 \cdot (N + P)^2}{g^2 \cdot (N - P)^2 + 4 \cdot P \cdot N} \geq 0.$$

The two solutions for this problem are

$$\rho = \frac{1 \pm \sqrt{D}}{2}.$$

As $\rho < 0.5$ must hold, only the minus returns a valid α.

Determining the DSPI: Suppose we have a realized $\overline{\text{MCC}}$ at our disposal. Solving Equation (5.33) for α yields

$$\begin{aligned}\alpha^2 \cdot ((\overline{\text{MCC}})^2 \cdot (\rho \cdot (N-P) + P - M \cdot \theta^*_{\text{opt}})^2 + P \cdot N \cdot (1-2\cdot\rho)^2) \\ -\alpha \cdot (\overline{\text{MCC}})^2 \cdot M \cdot (\rho \cdot (N-P) + P - M \cdot \theta^*_{\text{opt}}) \cdot (1 - 2 \cdot \theta^*_{\text{opt}}) \\ -M^2 \cdot (\overline{\text{MCC}})^2 \cdot \theta^*_{\text{opt}} \cdot (1 - \theta^*_{\text{opt}})) = 0.\end{aligned}$$

Solving this quadratic problem yields

$$\alpha = \frac{(\overline{\text{MCC}})^2 \cdot M \cdot (\rho \cdot (N-P) + P - M \cdot \theta^*_{\text{opt}}) \cdot (1 - 2 \cdot \theta^*_{\text{opt}}) \pm \sqrt{D}}{2 \cdot ((\overline{\text{MCC}})^2 \cdot (\rho \cdot (N-P) + P - M \cdot \theta^*_{\text{opt}})^2 + P \cdot N \cdot (1-2\cdot\rho)^2)},$$

with

$$\begin{aligned}D = (\overline{\text{MCC}})^4 \cdot M^2 \cdot (\rho \cdot (N-P) + P - M \cdot \theta^*_{\text{opt}})^2 \cdot (1 - 2 \cdot \theta^*_{\text{opt}})^2 \\ +4 \cdot F \cdot M^2 \cdot \theta^*_{\text{opt}} \cdot (1 - \theta^*_{\text{opt}}) \cdot (\overline{\text{MCC}})^2 \geq 0,\end{aligned}$$

where $F = (\overline{\text{MCC}})^2 \cdot (\rho \cdot (N-P) + P - M \cdot \theta^*_{\text{opt}})^2 + P \cdot N \cdot (1-2\cdot\rho)^2$. We can see that the $(-\sqrt{D})$ solution results in a negative α, so only the solution $(+\sqrt{D})$ should be valid. It is positive, but it is an open question to show that $\alpha = \frac{-b+\sqrt{D}}{2a} \in [0,1]$.

Scaling the DSPI: Equation (5.33) can be analyzed in its behavior by scaling the scores with a min-max scaler between μ_0 and μ_1 with the PP ($\rho = 0$). This gives us

$$\frac{\overline{\text{MCC}}|\rho = 0 - \mu_0}{\mu_1|\rho = 0 - \mu_0} = \overline{\text{MCC}}|\rho = 0 = \alpha \cdot \sqrt{\frac{P \cdot N}{E}},$$

where $E = -\alpha^2 \cdot (P - M \cdot \theta^*_{\text{opt}})^2 + \alpha \cdot M \cdot (P - M \cdot \theta^*_{\text{opt}}) \cdot (1 - 2 \cdot \theta^*_{\text{opt}}) + M^2 \cdot \theta^*_{\text{opt}} \cdot (1 - \theta^*_{\text{opt}})$.

6

ULTRA: Utilizing Transfer Learning to Overcome the Active Learning Cold-Start Phenomenon in Network Intrusion Detection

Contents

Based on [28]:
E.P. van de Bijl, S. Bhulai, and R.D. van der Mei, "ULTRA: Utilizing transfer learning to overcome the active learning cold-start phenomenon in network intrusion detection", submitted for publication.

Abstract

In *active learning*, models actively query the most informative data instances to reduce labeling effort. However, in a *cold-start setting*, the absence of labeled data prevents active learning strategies from effectively selecting relevant instances, often forcing reliance on random selection. This challenge is amplified in *intrusion detection* (ID) tasks, where class imbalance (an overwhelming presence of benign activities) makes it harder to identify malicious behavior. Publicly available ID datasets containing diverse attack types could be used to improve and accelerate an ID classifier's performance. Unfortunately, underlying network traffic distributions constantly change. As a consequence, traditional *machine learning* approaches struggle to generalize across old and new network data when this issue is not addressed. *Transfer learning* techniques offer a remedy against this so-called *domain adaptation* problem. Despite these techniques' promises, limited research has been done on effectively combining active learning and transfer learning to solve label scarcity in this domain. This work proposes a framework *utilizing transfer learning to overcome active learning's cold-start phenomenon* (ULTRA). Integrating both techniques in an iterative process, ULTRA improves the performance and robustness of ID classifiers when applied in new network environments. In summary, ULTRA overcomes: (1) *labeled data scarcity* when starting an active learning procedure, (2) *scalability issues* of feature-based transfer learning techniques, and (3) *negative transfer learning* by weighing labeled instances. We demonstrated the substantial advancements of ULTRA by comparing its performance in the cold-start situation on a set of four different ID datasets. ULTRA is a valuable tool for improving the efficiency and robustness of active learning classifiers utilized for ID.

6.1 Introduction

The rapid growth of the internet and the parallel increase in cyberattacks indicate the urgent need to improve detection techniques in cybersecurity. Traditional security measures, such as signature-based techniques, rely heavily on manually defined rules, but they are time-consuming to compose and maintain. In addition, they are ineffective against novel cyberattacks [105]. *Machine learning* (ML) has emerged as a promising alternative, offering adaptability and automation. ML classifiers rely on sufficient labeled data to make accurate predictions. However, this requirement is not always met when employing them in a new network environment.

Manually labeling network data to overcome this scarcity is time-intensive. The *active learning* (AL) paradigm offers a solution, enabling experts to focus only on iteratively labeling the most informative instances [15]. While this reduces the overall labeling effort, AL faces a critical obstacle at the outset: the *cold-start* phenomenon [22]. Data-dependent AL strategies require an initial set of labeled data to identify relevant instances for labeling, as it is otherwise reliant on random selection. This challenge is amplified in *intrusion detection* (ID) tasks, where network traffic is typically dominated by benign activities and malicious events are scarce, requiring an even greater volume of labeled data to build a classifier to distinguish them. Publicly available ID datasets that contain both network traffic types offer a potential remedy as supplementing data [23]. However, leveraging such datasets introduces a new challenge: the ever-changing nature of network traffic. Discrepancies between external datasets and real-world traffic can introduce unwanted bias or noise, reducing the classifier's prediction performance. This problem is typically addressed within the realm of *transfer learning* (TL), which focuses on extracting knowledge from one task (*source*) for the use of another (*target*) [18]. Specifically, in the *domain adaptation* (DA) setting, the challenge is to bridge the differing data distributions between *source* and *target* domains. Successfully navigating DA is critical to ensuring that external datasets enhance, rather than hinder, the performance of a classifier.

Recent research explores the integration of TL and AL for ID. However, some studies performing this integration are built on assumptions that lack practical justification. For example, Li *et al.* [106] assume that both source and target datasets contain unlabeled data, where the expert labels instances from the source dataset in an AL approach. This assumption overlooks a key reality: most publicly available ID datasets (source) are already labeled. Consequently, the focus of the AL process should shift towards labeling instances in the target dataset, where the expert's input is needed. Studies combining AL with TL often select one of the two primary TL techniques. On the one hand, there is *instance-based transfer*

learning (IBTL), where the objective is to select relevant source/target datasets instances for the target task [18]. The purpose of *negative transfer learning* (NTL) is to prevent including source instances irrelevant to the target task. Nonetheless, it requires sufficient target instances to distinguish relevant from irrelevant data [72]. On the other hand, *feature-based transfer learning* (FBTL) techniques aim at finding a *good* feature representation to minimize the domain divergence and classification error [18]. One issue in FBTL is that it overlooks the NTL issue, as all instances are considered equal when embedding the data. Another issue is *scalability*; finding the proper embedding can require heavy computational power to solve an optimization problem impacted by the size of the source and target datasets. To the best of the authors' knowledge, no research has explored the potential of combining these two TL techniques to complement each other and address their respective limitations in combination with AL in ID tasks.

In this chapter, we tackle this key problem by proposing a novel framework *utilizing TL to overcome AL's cold-start phenomenon* (ULTRA). Our method tackles three challenges. First, it combats AL's cold-start problem. With a DA technique called *semi-supervised transfer components analysis* (SSTCA) [107], [108], we utilize network data from other datasets and bridge distribution differences using this technique. Our improved version of SSTCA, which we call SSTCA$^+$, is not limited to only the labels from the source dataset but also incorporates acquired labeled instances from the target dataset. In this way, we can build a better model more quickly. Second, it combats the problem of NTL, which is the problem of selecting only relevant instances from another task for the task at hand. ULTRA gives higher weights to useful source instances, reducing the impact of those not useful for our problem and strengthening the classifier on target instances where the model makes mistakes. Third, it overcomes scalability issues of FBTL techniques by using IBTL techniques to select only a smaller set of relevant instances. We made the ULTRA code available in a repository [33]

This chapter is organized as follows: Section 6.2 reviews the relevant literature on AL and TL. Section 6.3 introduces the mathematical notation and discusses the components of the ULTRA method. Section 6.4 provides the mathematical notation for the AL procedure. In Section 6.5, we discuss the FBTL procedure used in ULTRA, while Section 6.6 provides how ULTRA integrates the IBTL procedure. In Section 6.7, we present the ULTRA algorithm in detail. Section 6.8 describes the experimental setup, while Section 6.9 reports the results. Finally, Section 6.10 discusses the insights, concludes the chapter, and outlines directions for future research.

6.2 Related Work

This section discusses related work and examines existing frameworks relevant to this research. We discuss their approaches, highlighting similarities and differences compared to our methodology.

6.2.1 Intrusion Detection

ID is the task of monitoring and detecting malicious activities or devices [109]. An *intrusion detection system* (IDS) can broadly be categorized as host-based (HIDS) or network-based (NIDS). A broad range of techniques to detect malicious events can be grouped into statistical-based, knowledge-based, or ML-based [7]. From a data perspective, ID datasets are packet-based or flow-based parsed [23]. This research focuses on ID in a NIDS setting using ML techniques applied to flow-based data. Still, we recognize the trade-offs inherent to this selection.

6.2.2 Active Learning

AL is a field within human-in-the-loop ML where experts, or 'oracles', label selected instances [14]. Labeling can be time-consuming, especially with large datasets, where labeling all instances may be unnecessary, redundant, or even introduce noise. AL addresses this by strategically selecting the most informative instance, enabling the construction of a high-accuracy classifier while minimizing the expert's labeling effort. Example studies researching AL in ID include [110]–[112]. Two primary strategies are used to query the oracle: *stream-based*, where each new instance is presented to the expert, and *pool-based*, where a set of candidate instances is given to the expert. We focus on pool-based sampling as selected instances from a pool are given to the cyber expert.

AL provides various model-dependent strategies to select the most relevant instances for labeling. An example is uncertainty sampling, as demonstrated in [16], where instances 'closest' to the decision boundary are prioritized. However, such strategies depend heavily on the classifier's ability to accurately model the decision boundary, which requires sufficient labeled data. Without this, the classifier may struggle to make reliable judgments, undermining the effectiveness of the selection process and the quality of the newly acquired data. When starting with little to no labels (i.e., a cold start), most active learners must resort to random sampling [113], [114]. This is also recognized by Klein *et al.* [115], where the percentage of randomly selected instances is adjusted based on the size of the initial labeled set. When the initial set is small, a higher proportion of random selection is used; as the labeled set grows, this proportion is reduced. This study utilizes TL techniques to overcome this cold-start phenomenon.

6.2.3 Transfer Learning

Traditional ML and data mining algorithms assume that the distribution of collected labeled and unlabeled training data are identical, but this assumption does not necessarily hold [18]. Within the study of TL, the aim is to develop algorithms to overcome this issue and overcome possible differences in the specific tasks or domains. Conceptually, it aims to extract knowledge from one or more *source tasks* and apply this knowledge to a (different) *target task*. In contrast to a multi-task learning approach, where both tasks are tackled simultaneously, we focus on performing the target task in this research.

Generally speaking, there are three techniques to perform TL. *Instance-based* TL aims at selecting relevant source/target instances to perform the target task. One of the first works on TL in the ID domain is performed by [116]. The issue with this approach is that it requires many samples from the target data to weigh instances accurately. *Parameter-based* TL is where parameters of a model are trained on one task and used on another task. One example of a parameter-based approach in ID is performed in Masum and Shahriar [76]. This study trains a convolutional neural network on network traffic from one dataset and tests on another. The issue is that network data is translated to images, which could result in information being lost due to this mapping. The last approach is the *feature-based* approach, where source and target datasets are projected to a 'good' feature representation. These methods are typically categorized into *heterogeneous* and *homogeneous* approaches. In the latter, it is assumed that the source and target data spaces are identical, whereas the former does not make this assumption. In the sequential works of Zhao *et al.* [72], [73], the authors focus on the assumption that network intrusion detection datasets are heterogeneous. In this research, as we work with flow-based ID, we assume that raw network traffic files can be converted into homogeneous datasets using merely one feature extractor. In Zhao *et al.* [72], data points are embedded into a manifold by finding a projection matrix that minimizes the distance between these points within the embedding space. The formulated optimization problem is similar to HeMap [117], but solved in a Lagrangian manner. In the later work, Zhao *et al.* [73] introduce a clustering step, where the first clusters of data points are created to preserve local geometry. These clusters are then used to embed the data using HeTL. Taghiyarrenani *et al.* [75] propose an embedding technique that leverages multiple datasets, deriving inspiration from the DAMA framework [118], which applies TL across multiple datasets.

6.2.4 Combining Learning Techniques

Let us now discuss four frameworks combining AL with TL and place them in the perspective of ULTRA.

ATL: Peng *et al.* [119] introduce the concept of *active transfer learning* (ATL). This method combines AL with TL by optimizing a weight vector for source instances. This is achieved through an iterative process involving the derivative of the Lagrangian optimization problem to refine the projection matrix, update Lagrangian multipliers, and improve both the projection matrix and the weighing vector. The critical distinction between this approach and ours is that their AL component focuses solely on selecting relevant source instances by refining their weight. In contrast, our approach extends this idea by applying AL to both source and target datasets while incorporating acquired labeled target data. However, the general concept of optimizing a weighing vector and a projection matrix aligns with our methodology.

ACTrAdaBoost: Similarly, Li *et al.* [106] explore the combination of AL and TL in the context of ID. Their AL procedure, called ACTrAdaBoost, labels instances in the source dataset to construct a refined source domain, assuming it contains unlabeled instances. In contrast, this study assumes a fully labeled source dataset, as we rely on publicly available datasets. Additionally, while their approach uses IBTL to tackle domain shift without labeling target instances, we adopt a different strategy. In our work, we actively label instances in the target dataset rather than the source dataset, addressing a more realistic scenario.

TrAcSVM: Zhu *et al.* [120] introduce TrAcSVM, an algorithm that combines AL with TL to enhance classification performance when labeled target data is scarce. Similar to the previously mentioned works, unlabeled instances from the source dataset are actively selected for labeling. However, TrAcSVM distinguishes itself by weighing and incorporating these newly labeled instances into embedding data into a new space. This hybrid approach integrates FBTL (via projection matrices) with instance-based transfer (through weighing), achieving robust DA while minimizing negative transfer. Our methodology shares similarities with this work, as we also reweigh instances to mitigate negative transfer and leverage the benefits of feature-based techniques.

ALTRA: Yuan *et al.* [16] introduce ALTRA, which inspired the naming of our method, ULTRA. ALTRA combines AL with IBTL to enhance learning in scenarios with limited labeled data. This approach first selects a subset of source data using a nearest-neighbor-based method. Subsequently, iteratively labeled instances are assigned weights following the TrAdaBoost algorithm Dai *et al.* [121], a method designed to handle TL by adjusting the influence of source and

target instances. Finally, ALTRA employs an (un)certainty decision rule to select instances, leveraging a weighted support vector machine to guide the selection process. While ALTRA effectively addresses the challenge of NTL by minimizing the impact of poorly matched source instances, it faces scalability challenges, particularly in the subset selection stage. Building on the foundational ideas of ALTRA, ULTRA incorporates an FBTL component into the framework. This integration aims to overcome the scalability limitations inherent in ALTRA while retaining its capacity to address NTL and ensure effective instance selection in TL scenarios.

6.3 Preliminaries

While the previous section provided an overview of AL and TL, this section formalizes these concepts with a mathematical framework.

6.3.1 Domain Adaptation

A *domain* $\mathcal{D}$, in the context of TL [108], [122], consists of two components: a k-dimensional feature space $\mathcal{X} \in \mathbb{R}^k$ and a marginal probability distribution $\mathcal{P}(\mathbf{X})$ where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subseteq \mathcal{X}$ represents a set of n instances. Formally, $\mathcal{D} \coloneqq \{\mathcal{X}, \mathcal{P}(\mathbf{X})\}$. The *source* and *target* domains differ whenever feature spaces or marginal probability distributions are not identical. For instance, this difference occurs due to *domain shift*. Such domain shift creates challenges for transferring models trained on the source domain to perform effectively on the target domain. For a domain, its corresponding *task* $\mathcal{T}$ is defined by two components: a label set $\mathcal{Y}$ and a classifier f. The classifier maps instances from the feature space to the label set, i.e., $f : \mathcal{X} \to \mathcal{Y}$. Formally, $\mathcal{T} \coloneqq \{f(\mathbf{x}), \mathcal{Y}\}$, where $\mathbf{x} \in \mathcal{X}$.

Let $\mathcal{X}_s$ and $\mathcal{X}_d$ denote the feature spaces of the *source* and *target* data, respectively. In this study, we assume identical feature spaces, placing our work within the domain of homogeneous transfer learning. We have a dataset $\mathbf{X}_s \coloneqq (\mathbf{x}_1^s, \mathbf{x}_2^s, \ldots, \mathbf{x}_n^s) \in \mathcal{X}_s$ consisting of n data samples from the *source* feature space and m data samples from the *target* feature space $\mathbf{X}_d \coloneqq (\mathbf{x}_1^d, \mathbf{x}_2^d, \ldots, \mathbf{x}_m^d) \in \mathcal{X}_d$. Say $\mathcal{P}_s(\mathbf{X}_s)$ and $\mathcal{P}_d(\mathbf{X}_d)$ are the marginal distributions over the *source* and *target* feature spaces. In many DA problems, a key assumption is that the marginal distributions differ, i.e., $\mathcal{P}_s \neq \mathcal{P}_d$, which highlights the necessity of adapting knowledge from the source domain to perform well in the target domain [108].

6.3.2 Road to ULTRA

As previously stated, AL and TL are often two distinctive fields, and limited work is performed on combining them. There is, however, a lot of benefit to incorporating these techniques [16], [22]. ULTRA is built upon a collection of existing work in these fields and created to integrate them to boost both methods. We will first discuss all components and combine them in the ULTRA algorithm. Figure 6.1 shows the order in which the specific components are stated. First, we introduce the AL notation, built upon the work in [115] and adjusted to the usage of TL. Second, the selected FBTL technique SSTCA [107], [108] is discussed and improved to a variant incorporating acquired labels by the AL framework. We call this new variant of this method SSTCA$^+$. Thirdly, we tackle the issue of NTL with a weighing update scheme extracted from TrAdaBoost [121], an IBTL technique. Mathematically, this method is enhanced by incorporating it into the AL framework. At last, we discuss the procedure of ULTRA iteratively using these components.

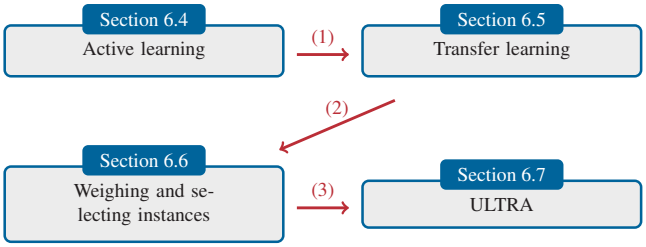


Figure 6.1: Roadmap to ULTRA

6.4 Active Learning

In ID, the classification task is to distinguish benign from malicious activities. Although many distinct classes of malicious events exist, we simplify this classification challenge by treating it as a binary problem, grouping all malicious events into one class. Say $\mathcal{Y} := \{0, 1, *\}$ is the set of possible labels. 0 here means that an instance is benign, while when it is 1, we say it is malicious. The third element $*$ indicates that the label is unknown. We assume that each instance $x_i^s \in \mathcal{X}_s$ has a corresponding label $y_i^s \in \{0, 1\}$. Let us denote $\mathbf{y}_s := (y_1^s, y_2^s, \ldots, y_n^s)$ as the vector containing the labels of the *source* data. The instances of the *target* dataset are unlabeled at our initial state. Let us denote $y_i^d(t)$ as the label of the i-th instance of the *target* dataset at AL iteration t. Let us denote $\mathbf{y}_d(t) = (y_1^d(t), y_2^d(t), \ldots, y_m^d(t))$ as the vector containing the labels of each instance at time-step t for $t \in \{1, 2, \ldots, T\}$. Here, T is the number of iterations that the

algorithm runs. We assume no data is labeled at our initial state, so we have $y_i^d(0) = *$ for $i \in 1, 2, \ldots, m$. We can combine both label vectors at time-step t as follows: $\mathbf{y}(t) \coloneqq (\mathbf{y}_s, \mathbf{y}_d(t))$.

Let us denote $\mathcal{L}_d(t)$ as the set of labeled *target* distribution instances at AL step t. At our initial state, we do not have any labeled data in the *target* dataset, so $\mathcal{L}_d(0) \coloneqq \emptyset$. $\mathcal{L}_s$ is the set of labeled instances of the *source* dataset. As this dataset is fully labeled, we do not need to label the source dataset; it is independent of step t. Mathematically, it is $\mathcal{L}_s \coloneqq \{(\mathbf{x}_1^s, y_1^s), (\mathbf{x}_2^s, y_2^s), \ldots (\mathbf{x}_n^s, y_n^s)\}$. The total labeled data at iteration t can be determined by $\mathcal{L}(t) \coloneqq \mathcal{L}_d(t) \cup \mathcal{L}_s$. Let us denote the indices at time-step t of the labeled data as follows: $\mathcal{I}_d(t) \coloneqq \{i \mid (x_i^d, y_i^d(t)) \in \mathcal{L}_d(t), i = 1, 2, \ldots, m\}$.

6

On the other hand, we denote the set of unlabeled instances at iteration t as $\mathcal{U}(t)$. At our initial time-step, we have $\mathcal{U}(0) \coloneqq \{(x_1^d, y_1^d(0)), (x_2^d, y_2^d(0)), \ldots, (x_m^d, y_m^d(0))\}$. In this notation, it appears that each instance is 'labeled' as each $y_i^d(0) = '*'$ for all $i \in 1, 2, \ldots, m$, but we use this as a placeholder to be correct in the formulation of the set notation.

In our procedure, we have an *expert* or *oracle* at our disposal to determine the label of an instance. At each iteration step t, we select $\mathcal{Q}(t)$ instances from our set of unlabeled instances $\mathcal{U}(t-1)$, subject to the constraint $\mathcal{Q}(t) \subseteq \mathcal{U}(t-1)$. Our expert, represented by function c, maps feature space $\mathcal{X}$ to $\mathcal{Y}$, assigning a label to each instance. The update rule for our labeling function at time-step $t = \{1, 2, \ldots, T\}$ for all *target* instances ($i \in 1, 2, \ldots, m$) is as follows:

$$y_i^d(t) \coloneqq \begin{cases} y_i^d(t-1) & \text{if } (\mathbf{x}_i^d, y_i^d(t-1)) \notin \mathcal{Q}(t), \\ c(\mathbf{x}_i^d) & \text{if } (\mathbf{x}_i^d, y_i^d(t-1)) \in \mathcal{Q}(t). \end{cases}$$

This updating scheme implies that once labeled, we assume that its label does not change: if $(\mathbf{x}_i^d, y_i^d(t)) \in \mathcal{L}_d(t)$, then $y_i^d(t) = y_i^d(t+1) = y_i^d(t+2) = \ldots$. Our set of queried samples $\mathcal{Q}(t)$ at time-step t must be updated before we add it to the set of labeled instances. Say $\mathcal{Q}'(t) \coloneqq \{(\mathbf{x}_i^d, y_i^d(t) \mid (\mathbf{x}_i^d, y_i^d(t-1)) \in \mathcal{Q}(t)\}$ is the set of updated labels. We can now add the updated labeled set to the set of labeled instances $\mathcal{L}_d(t) \coloneqq \mathcal{L}_d(t-1) \cup \mathcal{Q}'(t)$. After including these newly labeled instances in the set of labeled instances, we can remove the corresponding instances from our set of instances, that is, $\mathcal{U}(t) \coloneqq \mathcal{U}(t-1) \setminus \mathcal{Q}(t)$.

Labeling data in ID is often time-consuming; therefore, only a few instances can be labeled. Say $Q(t) \coloneqq |\mathcal{Q}(t)|$ is the size of the number of samples we query the oracle at time t. While the number of instances to be labeled can vary, this research assumes that the query pool size remains consistent across each iteration. Let us

denote this query size as $q = Q(t)$ for all $t \in \{1, 2, \ldots, T\}$. The total number of labeled instances in our procedure is Tq.

6.5 Embedding Source and Target Data

The AL framework, as described, incrementally provides new labeled instances from the target dataset. Since the source and target datasets originate from distinct networks, they exhibit notable distribution differences. ULTRA employs an FBTL technique called *semi-supervised transfer component analysis* (SSTCA) [107], [108] to tackle this. In this section, we explore the mathematics of SSTCA to provide insight into how its parameters influence the resulting embedding. In addition, we show how its improved version, which we named SSTCA$^+$, incorporates AL-acquired target instance labels in the mechanics. Before delving into the mechanisms of SSTCA, we introduce mathematical notation that selects a subset of all data to manage scalability issues in FBTL.

6.5.1 Instance Selection

Let us, for now, assume we have some mechanism, which we discuss later, to select $\check{n}$ instances from the source dataset and $\check{m}$ from the target dataset with $\check{n} \leq n$ and $\check{m} \leq m$. Mathematically, we have a subset of the source dataset $\mathcal{S}(t) \subseteq \mathcal{L}_s$ and a subset of the target dataset $\mathcal{T}(t) \subseteq \mathcal{L}_d(t) \cup \mathcal{U}_d(t)$. At each iteration t, we select (possibly) a different set of instances, making both sets iteration-dependent. This implies that the input data for the SSTCA algorithm is a subset of the whole dataset, so let us denote this subset as follows:

$$\mathbf{X}(t) \coloneqq \text{col}(\mathbf{X}_{\mathcal{S}(t)}, \mathbf{X}_{\mathcal{T}(t)}),$$

where col stacks vertically the matrices and $\mathbf{X}_{\mathcal{S}(t)} = (\mathbf{x}_1^{\mathcal{S}}(t), \mathbf{x}_2^{\mathcal{S}}(t), \ldots, \mathbf{x}_{\check{n}}^{\mathcal{S}}(t))$. As the feature spaces are identical, the stacked matrix has the same feature space. The label vector is given by $\mathbf{y}_{\mathcal{S}} = (y_1^{\mathcal{S}}(t), y_2^{\mathcal{S}}(t), \ldots, y_{\check{n}}^{\mathcal{S}}(t))$ and we combine both in $\mathcal{S}(t) = \{(\mathbf{x}_1^{\mathcal{S}}(t), y_1^{\mathcal{S}}(t)), (\mathbf{x}_2^{\mathcal{S}}(t), y_2^{\mathcal{S}}(t)), \ldots, (\mathbf{x}_{\check{n}}^{\mathcal{S}}(t), y_{\check{n}}^{\mathcal{S}}(t))\}$. The same steps can be performed for $\mathcal{T}(t)$. We assume that the subsets are of equal size in each iteration.

6.5.2 Distribution Matching

In FBTL, the goal is to map data points to a 'good' representation dataspace. A desirable mapping in our DA setting should strive to minimize the difference between source and target data distributions. In SSTCA, searching for this mapping is performed by trying to satisfy three different objectives. The first objective in SSTCA is to minimize the distance of the projected marginal distributions

$\mathcal{P}_s(\phi(X_S))$ and $\mathcal{P}_d(\phi(X_d))$, where ϕ is a universal kernel-induced feature map. A distance metric called the *maximum mean discrepancy* (MMD), designed by Gretton *et al.* [123], is selected in SSTCA and expresses the difference of the mean of the distribution computed in the *reproducing kernel Hilbert space* (RKHS). An advantage of using the MMD is that it is non-parametric and more robust against data dimensionality. The empirical estimation of the distance between the distribution of the sampled source and the target is given by

$$\text{MMD}(\mathbf{X}_{\mathcal{S}(t)}, \mathbf{X}_{\mathcal{T}(t)}) = \left\| \frac{1}{\check{n}} \sum_{i=1}^{\check{n}} \phi(\mathbf{x}_i^{\mathcal{S}}(t)) - \frac{1}{\check{m}} \sum_{j=1}^{\check{m}} \phi(\mathbf{x}_j^{\mathcal{T}}(t)) \right\|_{\mathcal{H}}^2, \tag{6.1}$$

where $\|\cdot\|_{\mathcal{H}}$ is the RKHS norm [123]. The closer this value is to zero, the more the distributions are similar, while higher values indicate greater differences. Equation (6.1) can be rewritten as a trace expression:

$$\text{MMD}(\mathbf{X}_{\mathcal{S}(t)}, \mathbf{X}_{\mathcal{T}(t)}) = \text{Tr}(\mathbf{KH}),$$

where the kernel $\mathbf{K} \in \mathbb{R}^{(\check{n}+\check{m})\times(\check{n}+\check{m})}$ is the kernel matrix with elements $k_{i,j} = \phi(\mathbf{x}_i)^\top \cdot \phi(\mathbf{x}_j)$. When selecting a linear kernel, the kernel matrix becomes $\mathbf{K}(t) = [\mathbf{X}(t)]^\top \mathbf{X}(t)$. As we select a different subset of $\mathbf{X}(t)$ each time-step t, we should write $\mathbf{K}(t)$ to be concise, but to make it more readable, we write $\mathbf{K}$. The matrix $\mathbf{H} \in \mathbb{R}^{(\check{n}+\check{m})\times(\check{n}+\check{m})}$ contains the coefficients to scale $\mathbf{K}$ according to Equation (6.1). Its elements are derived as follows:

$$h_{i,j} := \begin{cases} \frac{1}{\check{n}} & \text{if } i \leq \check{n} \text{ and } j \leq \check{n}, \\ \frac{1}{\check{m}} & \text{if } i > \check{n} \text{ and } j > \check{n}, \\ \frac{1}{\check{n}\cdot\check{m}} & \text{else.} \end{cases}$$

Let us consider a matrix $\mathbf{Z}$ that transforms the empirical kernel map features to an $\check{k}$ dimensional space (with $\check{k} \leq \check{n} + \check{m}$). Using this transformation matrix $\mathbf{Z}$ and kernel matrix $\mathbf{K}$ of the original input data, the kernel matrix of the mapped instances, called the *resultant* matrix, is given by $\mathbf{K}^* = \mathbf{KZZ}^\top\mathbf{K}$. The transformed data can be computed as $\mathbf{X}^* = \mathbf{KZ}$. The mathematical objective to minimize the MMD in the mapped instances is given as follows:

$$\text{MMD}(\mathbf{X}^*_{\mathcal{S}(t)}, \mathbf{X}^*_{\mathcal{T}(t)}) = \text{Tr}((\mathbf{KZZ}^\top\mathbf{K})\mathbf{H}) = \text{Tr}(\mathbf{Z}^\top\mathbf{KHKZ}). \tag{6.2}$$

Now that we have discussed the first objective of our optimization problem, let us continue to the second while keeping in mind that our approach simultaneously seeks to balance three objectives.

6.5.3 Preserving Data Properties and Matching Label Dependence

Merely focusing on minimizing the distance between the marginal distributions $\mathcal{P}_s(\phi(\mathbf{X}_s))$ and $\mathcal{P}_d(\phi(\mathbf{X}_d))$, as described in the first objective, risks disrupting relevant data properties essential for the target supervised learning task. To address this, Pan *et al.* [108] propose an additional objective in the optimization problem that aims to maximize the dependence between the embedding (represented by $\mathbf{K}^*$) and the available labeled data while also preserving variance in both the source and target domain data. Pan *et al.* [108] introduce the following label indicator matrix:

$$\mathbf{K}_Y^*(t) = \gamma \cdot \mathbf{K}_Y(t) + (1-\gamma) \cdot \mathbf{I}_{\check{n}+\check{m}}, \tag{6.3}$$

where $\mathbf{K}_Y(t) \in \mathbb{R}^{(\check{n}+\check{m})\times(\check{n}+\check{m})}$ is the kernel matrix computed on the labels available at time-step (t), which is only defined on the source domain, with $[k_Y]_{i,j}(t) = 1$ if $y_i(t) = y_j(t)$ and $(\cdot, y_i(t)), (\cdot, y_j(t)) \in \mathcal{L}_s(t)$ else 0. The first term in Equation (6.3) controls the label dependence, while the second aims to maximize data variance. $\gamma \in [0,1]$ is a trade-off parameter describing the weight of focusing on either.

In contrast to the original formulation of SSTCA, we also iteratively retrieve labels from the target data. This means that we have more information and can update the indicator matrix. The elements in the kernel matrix for the labels in this scenario are given by $[k_Y]_{i,j}(t) = 1$ if $y_i(t) = y_j(t)$ and it holds that $(\cdot, y_i(t)), (\cdot, y_j(t)) \in \mathcal{L}(t)$, and zero otherwise. To differentiate this improvement from the original algorithm, we call this method SSTCA$^+$. To make the notation more convenient, we ignore the notation (t) from now on when describing $\mathbf{K}_Y^*$.

The dependency between the two matrices in SSTCA is measured with the *Hilbert space independence criterion* (HSIC) [124], a robust, nonparametric method for quantifying dependence between the embedding and the label indicator matrix. Its empirical estimate of the dependency between the two matrices is computed as follows:

$$\text{HSIC}(\mathbf{X}^*_{\mathcal{S}(t)}, \mathbf{X}^*_{\mathcal{T}(t)}) \coloneqq \frac{1}{(\check{n}+\check{m}-1)^2} \operatorname{tr}(\mathbf{C}\mathbf{K}^*\mathbf{C}\mathbf{K}_Y^*),$$

where $\mathbf{C} = \mathbf{I} - \frac{1}{\check{n}+\check{m}}\mathbf{1}\mathbf{1}^\top$ is the centering matrix. An HSIC value of zero indicates independence, and stronger dependence is reflected in higher values. When dropping the scaling factor, the following trace problem is maximized:

$$\text{Tr}(\mathbf{C}\mathbf{K}^*\mathbf{C}\mathbf{K}_Y^*) = \text{Tr}(\mathbf{C}(\mathbf{K}\mathbf{Z}\mathbf{Z}^\top\mathbf{K})\mathbf{C}\mathbf{K}_Y^*) = \text{Tr}(\mathbf{Z}^\top\mathbf{K}\mathbf{C}\mathbf{K}_Y^*\mathbf{C}\mathbf{K}\mathbf{Z}). \tag{6.4}$$

Now that the second objective value is discussed, we continue to the third objective for the optimization problem of SSTCA.

6.5.4 Locality Preservation

The final objective of the SSTCA optimization problem ensures the preservation of the local geometry properties of data instances within the embedding. Instances close to each other in the input space should remain close when projected. The manifold regularizer proposed in Belkin *et al.* [125] offers a means to satisfy this property. Let us denote $\mathcal{N}$ as the set of instance pairs where they are in the original feature space their r-nearest neighbor. Say $\mathbf{A} \in \mathbb{R}^{(\check{n}+\check{m})\times(\check{n}+\check{m})}$ is the affinity matrix, and its element-wise values are computed as follows:

$$a_{ij} \coloneqq \begin{cases} e^{\left(\frac{[\|x_i - x_j\|]^2}{2\cdot\sigma^2}\right)} & \text{if } (x_i, x_j) \in \mathcal{N}, \\ 0 & \text{else,} \end{cases}$$

where $\sigma > 0$ is a scaling parameter. The Laplacian matrix encapsulates the geometry of the data manifold. This Laplacian matrix $\mathbf{L} \in \mathbb{R}^{(\check{n}+\check{m})\times(\check{n}+\check{m})}$ is constructed as $\mathbf{L} \coloneqq \mathbf{D} - \mathbf{A}$ where $\mathbf{D}$ is the diagonal degree matrix with entries $d_{i,i} \coloneqq \sum_{g=1}^{\check{n}+\check{m}} a_{i,g}$ for $i \in \{1, 2, \ldots, \check{n} + \check{m}\}$. Now, say x_i^* is the i-th row of the matrix $\mathbf{KZ}$, the embedded data point. The objective, as stated in Matasci *et al.* [107] and Pan *et al.* [108], to be minimized becomes

$$\frac{1}{(\check{n}+\check{m})^2} \sum_{i,j} a_{i,j} \|x_i^* - x_j^*\|^2 = \frac{1}{(\check{n}+\check{m})^2} \mathrm{Tr}(\mathbf{Z}^\top \mathbf{KLKZ}). \tag{6.5}$$

Now that all three objectives have been discussed, we combine them in the next section in an optimization problem to find the projection matrix.

6.5.5 Optimization Problem

In the previous sections, we described the objectives SSTCA aims to achieve. Combining the terms of Equations (6.2), (6.4), and (6.5) leads to the following optimization problem:

$$\arg\min_{\mathbf{Z}} \mathrm{Tr}(\mathbf{Z}^\top \mathbf{KHKZ}) + \mu \cdot \mathrm{Tr}(\mathbf{Z}^\top \mathbf{Z}) + \frac{\lambda}{(\check{n}+\check{m})^2} \mathrm{Tr}(\mathbf{Z}^\top \mathbf{KLKZ}), \tag{6.6}$$

$$\text{s.t. } \mathbf{Z}^\top \mathbf{KCK}_Y^* \mathbf{CKZ} = \mathbf{I}_{\check{n}+\check{m}}.$$

Here, $\mu > 0$ and $\lambda > 0$ are trade-off parameters. As λ is a scaling factor, we can ignore the term $\frac{1}{(\check{n}+\check{m})^2}$. A standard regularization term $\mathrm{Tr}(\mathbf{Z}^\top \mathbf{Z})$ is utilized to prevent transformation matrix $\mathbf{Z}$ from overfitting. It also helps avoid rank deficiency of the denominator in the generalized eigenvalue decomposition. In this formulation, the data variance preservation property is enforced as a constraint, while the objective function minimizes the remaining objectives. If we set $\gamma = 0$ and

$\lambda = 0$, we get the non-supervised variant of SSTCA, called TCA. The optimization problem (6.6) can be reformulated as a generalized eigenproblem [108]. The optimal $\mathbf{Z}$ solving this problem can be constructed by combining the $\check{k} \leq \check{n} + \check{m}$ leading eigenvectors of the matrix

$$(\mathbf{K}(\mathbf{H} + \lambda\mathbf{L})\mathbf{K} + \mu\mathbf{I})^{-1}\mathbf{K}\mathbf{C}\mathbf{K}_Y^*\mathbf{C}\mathbf{K}.$$

As $(\mathbf{K}(\mathbf{H}+\lambda\mathbf{L})\mathbf{K}+\mu\mathbf{I})$ is non-singular, the matrix is invertible and thus an inverse can be derived [108]. The next section describes how the obtained $\mathbf{Z}$ transforms the data into the embedded space.

6.5.6 Projection Matrix

The transformation of new data points, using a selected kernel function $\mathcal{K}$, into the manifold can be achieved using the following mapping:

$$\upsilon(\mathbf{x}) \coloneqq \mathcal{K}(\mathbf{x}, \mathbf{X}(t)) \cdot \mathbf{Z},$$

where $\mathbf{Z}$ is the kernel-adjusted projection matrix obtained from the SSTCA procedure. The data is embedded by constructing a kernel matrix over the data and multiplying it with the projection matrix $\mathbf{Z}$. Specifically, when using a linear kernel, the embedding process simplifies to $\mathbf{X} \cdot [\mathbf{X}(t)]^\top \cdot \mathbf{Z}$. We can combine the last two matrix multiplications in a single matrix to streamline computation:

$$\mathbf{P} \coloneqq [\mathbf{X}(t)]^\top \cdot \mathbf{Z}.$$

To systematically map entire sets of instances into the embedded space, we define the following operator for any set $\mathcal{S}$ of data points:

$$\varphi(\mathcal{S}, \mathbf{P}) \coloneqq \{(\mathbf{x} \cdot \mathbf{P}, y) \mid \forall(\mathbf{x}, y) \in \mathcal{S}\}.$$

This operator maps each instance in the set $\mathcal{S}$ into the new feature space defined by the projection matrix $\mathbf{P}$, preserving the associated labels y. Given that our AL procedure partitions instances into labeled and unlabeled sets, the above mapping function is essential for transforming all data instances into the embedded space during each iteration. Still, an open question is how to select a representable subset of the data to create this projection matrix.

6.6 Selecting and Weighing Instances

Having covered how data is projected into a new representation for the ID task, this section focuses on ULTRA's selection mechanism to select a representable subset of the data to create the projection matrix $\mathbf{P}$. This section discusses how relevant instances are identified and how less relevant ones are less likely to be selected.

6.6.1 Loss Function Vector

After constructing the projection matrix $\mathbf{P}$, we embed instances to the new manifold and fit a classifier f. This classifier can return either binary predictions or confidence in malicious scores between 0 and 1. To evaluate the classifier's performance in distinguishing benign from malicious events, we compute the *loss function vector*, quantifying discrepancies between predictions and actual labels. For the source instances, the loss for a given projection matrix $\mathbf{P}$ and a classifier f is defined as

$$l_i^s(\mathbf{P}, f) \coloneqq |f(\mathbf{x}_i^s \cdot \mathbf{P}) - y_i^s|,$$

for $i \in 1, 2, \ldots, n$,where $\mathbf{x}_i^s, y_i$ is the i-th instance with its corresponding label. The loss depends on whether labels are already available for the target instances. If an instance is labeled at AL step t, its loss is calculated similarly for the source instances. Otherwise, the loss is set to zero to exclude its influence

$$l_j^s(\mathbf{P}, f, t) \coloneqq \begin{cases} |f(\mathbf{x}_j^d \cdot \mathbf{P}) - y_j^d(t)| & \text{if } j \in \mathcal{I}_d(t), \\ 0 & \text{if } j \notin \mathcal{I}_d(t), \end{cases}$$

for $j \in 1, 2, \ldots m$, where $\mathcal{I}_d(t)$ is the set of indices of labeled target instances at step t. If f returns confidence values, the loss lies in $[0, 1]$, while if it produces binary predictions, the loss takes values in $\{0, 1\}$.

6.6.2 Weighing Instances

While aligning the source and target network intrusion data via an embedding, we continuously reevaluate the utility of the embedded instances in the projected space. Each instance is assigned a weight, updated during each AL iteration by considering the instance's prediction and its actual label. Our weighing scheme is built upon the update mechanism of TrAdaBoost [121].

Let the weight vector, representing the weights for all instances, be denoted as $\mathbf{w}(t) \coloneqq (\mathbf{w}^s(t), \mathbf{w}^d(t))$ where $\mathbf{w}^s(t) \coloneqq (w_1^s(t), w_2^s(t), \ldots, w_n^s(t))$ and $\mathbf{w}^d(t) \coloneqq (w_1^d(t), w_2^d(t), \ldots, w_m^d(t))$. The model error for a classifier f applied on the projected data (with projection matrix $\mathbf{P}$) using the weights from previous iteration $w(t-1)$, is defined as

$$\epsilon(\mathbf{P}, f, t) \coloneqq \sum_{i \in \mathcal{I}_d(t)} \frac{w_i^d(t-1) \cdot l_i^d(\mathbf{P}, f, t)}{\sum_{j \in \mathcal{I}_d(t)} w_j^d(t-1)}. \tag{6.7}$$

Here, $l_i^d(\mathbf{P}, f, t)$ is the loss of a target instance i at iteration step t, as described in the previous section. Since the loss is normalized over the total weight, $\epsilon(\mathbf{P}, f, t)$ is always in the range $[0, 1]$. Notably, the weights from the previous iteration are

used because the weights are updated after computing the error. For simplicity, let us denote the error $\epsilon(\mathbf{P}, f, t)$ as $\epsilon(t)$. The update factor is computed as follows:

$$\beta_d(t) \coloneqq \begin{cases} 1, & \text{if } \epsilon(t) = 0 \text{ or } \epsilon(t) > 0.5, \\ \frac{\epsilon(t)}{1-\epsilon(t)}, & \text{if } 0 < \epsilon(t) \leq 0.5. \end{cases}$$

This update rule ensures that the weight remains unchanged for prediction situations where the error is either worse than a random prediction or when perfect predictions are achieved. The source data gets a different update factor; since it depends on its size and the total number of AL iterations, it is given by

$$\beta_s \coloneqq \frac{1}{1 + \sqrt{2 \cdot \ln(n)/T}}.$$

The weight update for the source instances at time-step $t \in \{1, 2, \ldots, T\}$ is given by

$$w_i^s(t) \coloneqq w_i^s(t-1) \cdot (\beta_s)^{l_i^s(\mathbf{P}(t), f, t)}$$

for $i = 1, 2, \ldots, n$. With this weight update, source instances where the classifier f makes wrong predictions are lowered to decrease their influence on the projection matrix. For the target data, the opposite is true: weights of wrongly predicted instances are increased to strengthen the importance of these instances. Only instances that are labeled are updated at time-step $t \in \{1, 2, \ldots, T\}$ using

$$w_i^d(t) \coloneqq \begin{cases} w_i^d(t-1) \cdot (\beta_d(t))^{-l_i^d(\mathbf{P}(t), f, t)} & \text{if } i \in \mathcal{I}_d(t), \\ w_i^d(t-1) & \text{if } i \notin \mathcal{I}_d(t), \end{cases}$$

for $i = 1, 2, \ldots, m$. In this framework, f is the classifier trained on the projected data, and the weights for unlabeled instances are retained from the previous iteration until new labels are obtained.

6.6.3 Instance Weight Distribution

In the previous section, we discussed the determination of the weight vector and its updating rule for the AL process. The current weight vector, however, is not normalized or differentiated between labeled and unlabeled instances. Let the weight distribution across instances at step t be represented as

$$\tilde{\mathbf{w}}(t) = (\tilde{\mathbf{w}}_s(t), \tilde{\mathbf{w}}_d(t)),$$

where $\tilde{\mathbf{w}}_d(t) \coloneqq (\tilde{w}_1^d(t), \tilde{w}_2^d(t), \ldots, \tilde{w}_m^d(t))$ and $\tilde{\mathbf{w}}_s(t) \coloneqq (\tilde{w}_1^s(t), \tilde{w}_2^s(t), \ldots, \tilde{w}_n^s(t))$. To normalize the weights, we first compute the total weight of the labeled

instances at time-step $t \in \{0, 1, 2, \ldots, T\}$, given by

$$v(t) \coloneqq \sum_{i=1}^{n} w_i^s(t) + \sum_{j \in \mathcal{I}_d(t+t)} w_j^d(t).$$

With this value, we can compute for the labeled instances at time-step $t \in \{0, 1, 2, \ldots, T\}$ the normalized values as follows: $\tilde{w}_i^s(t) \coloneqq \frac{w_i^s(t)}{v(t)}$, for $i = 1, 2, \ldots, n$ and for all target instances whether they are labeled or not:

$$\tilde{w}_i^d(t) \coloneqq \begin{cases} \frac{w_i^d(t)}{v(t)} & \text{if } i \in \mathcal{I}_d(t), \\ 0 & \text{if } i \notin \mathcal{I}_d(t), \end{cases}$$

for $i = 1, 2, \ldots, m$.

6.6.4 Subset Selection Algorithm

The normalized weight vector $\tilde{\mathbf{w}}(t-1)$ is used to select the most relevant instances for constructing the projection matrix. We prioritize instances with the highest normalized weights for the source dataset, as these are most representative of the target data. Priority is given to labeled instances for the target dataset, as their weights have been updated. We prefer using these instances for the projection matrix. The subset algorithm selects at time-step t the $\check{n}$ instances with their highest normalized weights $\tilde{\mathbf{w}}_s(t-1)$ from the source dataset to form $\mathcal{S}(t)$ as indicated in Section 6.5.1. In the same manner, we select $\check{m}$ instances using $\tilde{\mathbf{w}}_d(t-1)$ from the target dataset to form $\mathcal{T}(t)$. In the case of ties, the selection is performed randomly. This strategy ensures that the projection matrix incorporates the most informative labeled instances from the source and target datasets, optimizing alignment between them.

6.7 ULTRA

Algorithm 1 presents the ULTRA algorithm, combining the AL and TL techniques to overcome model instability in the cold-start AL setting. An oracle selects and labels a random batch of instances in each iteration, expanding the labeled target set. The algorithm then computes a weight vector $\tilde{\mathbf{w}}$, which normalizes the contributions of the labeled instances. This vector is used to guide the selection of a subset of both source and target data, ensuring that relevant samples contribute to creating the projection matrix $\mathbf{P}$. The selected data is then used to construct a projection matrix $\mathbf{P}$ via SSTCA, embedding the data into a new feature space that aligns the source and target distributions. Once the data is projected, a learner f is

trained on the transformed labeled set. The performance of f is monitored using error calculations, which help assess the relevance of labeled instances and adjust their weighting accordingly. This iterative refinement continues until the stopping criterion is met, allowing ULTRA to incrementally improve classification performance despite the initially limited availability of labeled target data.

Algorithm 1 Pseudocode ULTRA

Require:
Source dataset $\mathbf{X}_s$, source labels $\mathbf{y}_s$, target dataset $\mathbf{X}_d$, stopping iteration T, query size q

Initialize: $\mathcal{U}(0)$, $\mathcal{L}_s$, $\mathcal{L}_d(0)$, $\mathcal{L}(0)$, $\mathbf{y}(0)$, $\mathbf{w}(0)$
$\mathbf{P}(0) \coloneqq \mathbf{I}_k$ # Projection matrix
for $t = 1$ to T **do**
    $\mathcal{Q}(t) \Leftarrow$ Select randomly q instances from $\mathcal{U}(t-1)$
    Determine $\mathbf{y}(t)$ by labeling the set $\mathcal{Q}(t)$ by human expert (Oracle)
    Compute $\mathcal{Q}'(t)$ with the updated labels
    Update $\mathcal{L}_d(t)$, $\mathcal{L}(t)$ and $\mathcal{U}(t)$
    Determine $\tilde{\mathbf{w}}(t-1)$
    $\mathcal{S}(t) \Leftarrow$ subset($\mathcal{L}_s$, $\tilde{\mathbf{w}}_s(t-1)$, $\check{n}$)
    $\mathcal{T}(t) \Leftarrow$ subset($\mathcal{L}_d(t) \cup \mathcal{U}(t)$, $\tilde{\mathbf{w}}_d(t-1)$, $\check{m}$)
    $\mathbf{P}(t) \Leftarrow SSTCA(\mathcal{S}(t), \mathcal{T}(t))$
    Train **Learner** f on $\varphi(\mathcal{L}(t), \mathbf{P}(t))$
    Determine loss vector $\mathbf{l}(\mathbf{P}(t), h, t)$
    Calculate error rate $\epsilon(\mathbf{P}(t), h, t)$ of f on $\varphi(\mathcal{L}_d(t), \mathbf{P}(t))$ by Equation (6.7)
    Determine updated $\mathbf{w}(t)$
end for

6.8 Experiments

In this section, we discuss the research's experiments. First, we elaborate on how we measure a classifier's performance. Second, we state the selected datasets. Third, we discuss this study's experimental setup and how we gather results. At last, we discuss the chosen classifiers for the binary classification task.

6.8.1 Performance Quantification

The performance of a binary classifier is measured by comparing the labels of predicted instances, denoted by $\hat{\mathbf{y}}$, with their actual labels. Let us denote, for our binary classification problem, the four base measures as follows: the number of *true positives* (TP), *true negatives* (TN), *false negatives* (FN), and *false positives*

(FP). We define $\hat{P} = \text{TP} + \text{FP}$ as the sum of the predicted positive instances and $\hat{N} = \text{TN} + \text{FN}$ as the sum of the predicted negatives. Performance metrics are higher-order evaluation quantifiers derived from these base measures. We have selected the *Matthews' correlation coefficient* (MCC) [55] metric to assess a model's performance. To perform a sanity check on the performance, we also computed the Dutch Draw (DD) baseline, see Chapters 2 and 3, for the metric to check if the selected classifiers outperform this simple and essential baseline. For the MCC, all DD classifiers (without the all-one or all-zero classifier) have an expected value of 0.0, meaning the DD baseline is 0.0, the result of the best random input-independent binary classifier. In addition to the chosen performance metrics, we want to determine how much the models have learned by computing the *Dutch Scaler performance indicator* (DSPI), as describes in Chapter 5. This performance indicator expresses quantitatively how much a model has learned.

6.8.2 Datasets

Proposed NID methods are traditionally evaluated on datasets such as the NSL-KDD [126] or Kyoto2006+ [127]. These datasets are, however, heavily outdated and do not represent current network behavior [23]. We require more recent datasets with the same data space as we focus on homogenous TL. Therefore, we have selected the UNSW-NB15 [81], CIC-IDS-2018 [77], BoT-IoT [128], and ToN-IoT [129] datasets provided in a uniform dataspace format (NetFlow V1) [130].

6.8.3 Experimental Setup

Publicly available datasets often exhibit significant class imbalance. We applied stratified sampling to adjust the class distributions to create a more realistic and meaningful experimental setup. For the source dataset, we balanced the benign and malicious classes with a ratio of 0.5. In contrast, the target dataset was designed to reflect real-world class skewness by setting the benign-malicious ratio to 0.05. Finally, the evaluation dataset was again balanced at a 0.5 ratio, allowing us to assess the classifier's ability to distinguish between the two classes under unbiased conditions.

6.8.4 Classification Models

There is a wide range of supervised learning models available for binary classification. *Support vector machines* (SVM) are frequently used in TL studies for evaluation purposes [73], [107], [108], [121]. However, SVM models are complex and often require enhancement in the ID domain before outperforming other models [8]. On the other hand, tree-based models are more straightforward and

require less implementation. To this end, we selected two tree-based models, *decision tree* (DT) and *random forest* (RF), a *nearest-neighbor classifier* (NN), and an SVM model to evaluate the results for ULTRA. When applying these models in the experiments, we selected the default scikit-learn hyperparameters [49].

6.9 Results

This section presents the results of the experiments conducted in this study. Table 6.1 gives an overview of the parameters selected to run ULTRA. As our focus is to overcome the cold-start problem in AL, the AL strategy is to select random instances from the target dataset. As a result, only the query size and the number of AL iterations need to be chosen. The remaining parameters pertain to the (SS)TCA($^+$) algorithms used within ULTRA. We begin by analyzing the parameter selection for the TL algorithm. Subsequently, we present the performance results of ULTRA with the selected parameters.

Table 6.1: Parameters of ULTRA and their description

Parameter	Description
μ	Trade-off parameter regularization
γ	Trade-off parameter data variance versus label alignment
λ	Trade-off parameter local preservation
r	Number of neighbors local preservation
σ	Scaling factor affinity matrix
$\check{k}$	Number of eigenvectors/ data dimensionality projected space
$\check{n}$	Number of selected source instances for SSTCA
$\check{m}$	Number of selected target instances for SSTCA
T	Number of AL iterations
q	Query size

6.9.1 Analysis of (SS)TCA($^+$) Parameters

The projection matrix obtained within the (SS)TCA($^+$) optimization problem depends on the parameter values described in Section 6.5. Matasci *et al.* [107] and Pan *et al.* [108] assume that $\mu = 1, \gamma = 0.5$ are the standard values for the optimization problem. However, the optimal values for the other variables are not precisely specified and should be studied. To study the effect of the parameters, we vary the values of the parameters and look at the impact on the objective value and the algorithm's performance on a test set. We study the effect of the paramet-

ers in this section with the NN classifier to prevent introducing more stochasticity in the outcomes. First, let us look at the parameter $\check{k}$ determining the number of components selected in (SS)TCA($^+$). We perform this analysis using the TCA variant to prevent other parameters from influencing the results. We randomly sampled $1,000$ source, target, and evaluation data instances in the manner as described in Section 6.8.3. Table 6.2 shows the performance of the NN algorithm when applied to TCA-transformed data in two situations: when no target data is labeled and when 100 random instances are labeled. The NN classifier is trained on the combined labeled dataset to study the effect of the number of components. The results indicate that performances are almost identical but vary when selecting two components.

Table 6.2: MCC scores of NN classifier (trained on all labeled data) on TCA embedded data with varying $\check{k}$

		# instances labeled $\lvert\mathcal{L}_d\rvert$, # components selected $\check{k}$							
Source	**Target**	**0, 2**	**0, 4**	**0, 6**	**0, 8**	**100, 2**	**100, 4**	**100, 6**	**100, 8**
BoT-IoT	CIC-IDS-2018	0.005	0.003	0.005	0.004	0.458	0.458	0.458	0.457
	ToN-IoT	0.057	0.057	0.058	0.058	0.520	0.521	0.521	0.521
	UNSW-NB15	0.106	0.097	0.097	0.097	0.481	0.481	0.481	0.481
CIC-IDS-2018	BoT-IoT	0.165	0.165	0.165	0.165	0.490	0.506	0.507	0.573
	ToN-IoT	-0.084	-0.009	-0.008	-0.002	0.468	0.486	0.485	0.487
	UNSW-NB15	0.350	0.338	0.339	0.339	0.483	0.492	0.492	0.492
ToN-IoT	BoT-IoT	0.184	0.185	0.185	0.185	0.737	0.738	0.738	0.738
	CIC-IDS-2018	0.047	0.128	0.101	0.101	0.445	0.573	0.572	0.575
	UNSW-NB15	-0.026	-0.045	-0.045	-0.045	0.362	0.367	0.367	0.367
UNSW-NB15	BoT-IoT	-0.084	-0.084	-0.084	-0.084	0.729	0.731	0.731	0.731
	CIC-IDS-2018	0.569	0.628	0.628	0.627	0.747	0.747	0.748	0.748
	ToN-IoT	-0.041	-0.041	-0.041	-0.041	0.447	0.447	0.447	0.447

Three connected parameters concerning the locality preservation of data properties are σ, λ, and r. All three penalize data distances in the projected space differently. Figure 6.2 shows the effect of σ on the objective value of the optimization problem. The data points are generated using the SSTCA and SSTCA$^+$ optimization versions and the four selected ML classifiers. Source, target, and evaluation datasets comprise $1,000$ instances, selected as discussed in Section 6.8.3. Three values are considered for σ: the mean and median of the feature distances in the projected space and simply 1. The mean and median distance change as different instances are selected for each active learning iteration. It can be observed that taking the median/mean results in higher objective values. Still, it does not necessarily result in a projection matrix strengthening the performance of the classifiers for this source and target dataset combination.

Next, we examine the results of the NN classifier when varying the parameter λ.

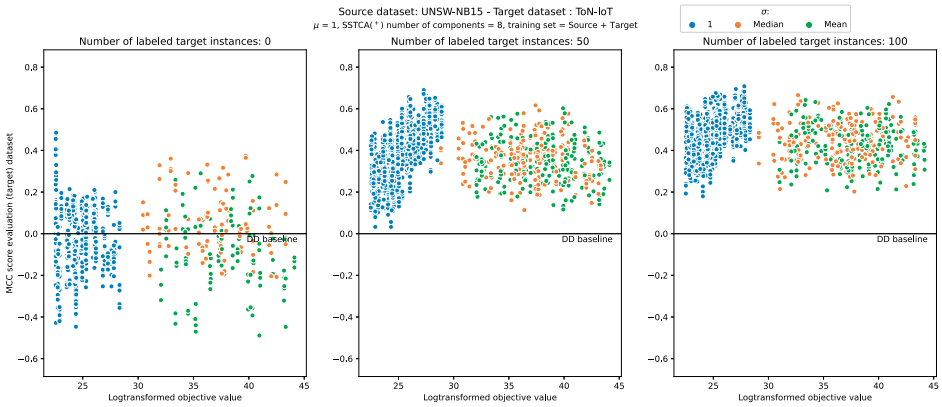


Figure 6.2: Effect of σ on the objective value and classifier performance

We considered three values for λ: $\{1, 100, 1, 000\}$. Figure 6.3 illustrates how increasing λ affects the objective values of the SSTCA($^+$) optimization problem. As λ increases, the objective values decrease. However, as shown in Table 6.3, a lower objective value does not necessarily translate into better classifier performance. Additionally, when studying the effect of the number of neighbors (r), evaluated at $\{50, 100, 200\}$, we find no clear relationship between this parameter and either the objective function or performance outcomes.

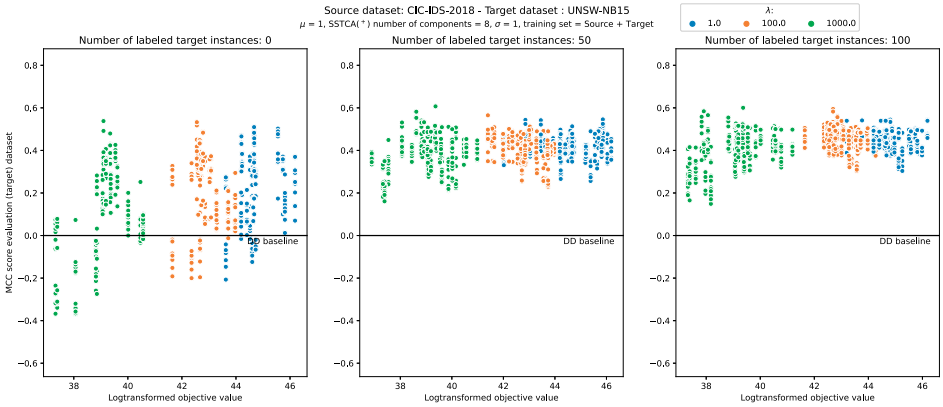


Figure 6.3: Effect of λ on the objective value and classifier performance

We now compare the performance of the NN classifier applied to the embeddings generated by the TCA, SSTCA, and SSTCA$^+$ algorithms. For this comparison, we set $\sigma = 1$, $\lambda = 1$, and $r = 100$. Table 6.4 reports the average MCC scores of the NN classifier for the various embeddings. The highest values for each source and target dataset combination are highlighted in bold, considering two scenarios:

Table 6.3: MCC scores of NN classifier (trained on all labeled data) on SSTCA($^+$) embedding with varying λ

		# instances labeled $\lvert\mathcal{L}_d\rvert$, λ					
Source	**Target**	**0, 1**	**0, 100**	**0, 1,000**	**100, 1**	**100, 100**	**100, 1,000**
BoT-IoT	CIC-IDS-2018	-0.051	0.052	0.195	0.532	0.558	0.634
	ToN-IoT	0.086	0.121	0.143	0.526	0.517	0.473
	UNSW-NB15	-0.110	-0.227	-0.247	0.467	0.462	0.497
CIC-IDS-2018	BoT-IoT	0.112	0.136	0.225	0.562	0.514	0.580
	ToN-IoT	-0.230	-0.132	-0.060	0.468	0.451	0.455
	UNSW-NB15	0.139	0.035	0.077	0.493	0.486	0.505
ToN-IoT	BoT-IoT	-0.002	-0.062	-0.075	0.709	0.707	0.714
	CIC-IDS-2018	0.205	0.166	0.024	0.543	0.532	0.544
	UNSW-NB15	-0.011	-0.022	0.093	0.404	0.387	0.447
UNSW-NB15	BoT-IoT	-0.030	0.082	-0.075	0.734	0.730	0.732
	CIC-IDS-2018	0.642	0.639	0.640	0.779	0.779	0.779
	ToN-IoT	-0.072	-0.041	-0.076	0.475	0.449	0.482

one with no labeled target data and another where 100 randomly selected target instances are labeled. When no target instances are labeled, SSTCA and SSTCA$^+$ perform identically, as they have the same amount of knowledge. However, when a small subset of target instances is labeled, SSTCA$^+$ outperforms SSTCA, particularly when UNSW-NB15 is used as the source dataset.

Table 6.4: Average MCC scores of an NN classifier applied on raw and embedded data

		Number of instances labeled $\lvert\mathcal{L}_d\rvert$, Data space							
Source	**Target**	**0, Raw**	**0, SSTCA**	**0, SSTCA$^+$**	**0, TCA**	**100, Raw**	**100, SSTCA**	**100, SSTCA$^+$**	**100, TCA**
BoT-IoT	CIC-IDS-2018	**0.023**	-0.051	-0.051	0.004	**0.613**	0.537	0.526	0.457
	ToN-IoT	**0.361**	0.086	0.086	0.058	0.464	**0.528**	0.524	0.521
	UNSW-NB15	-0.146	-0.110	-0.110	**0.097**	0.467	0.468	0.467	**0.481**
CIC-IDS-2018	BoT-IoT	**0.204**	0.112	0.112	0.165	0.515	**0.592**	0.531	0.573
	ToN-IoT	-0.092	-0.230	-0.230	**-0.002**	0.412	0.470	0.466	**0.487**
	UNSW-NB15	0.103	0.139	0.139	**0.339**	0.486	**0.493**	**0.493**	0.492
ToN-IoT	BoT-IoT	-0.095	-0.002	-0.002	**0.185**	**0.744**	0.708	0.710	0.738
	CIC-IDS-2018	0.055	**0.205**	**0.205**	0.101	**0.595**	0.515	0.572	0.575
	UNSW-NB15	-0.199	**-0.011**	**-0.011**	-0.045	**0.415**	0.406	0.403	0.367
UNSW-NB15	BoT-IoT	-0.068	**-0.030**	**-0.030**	-0.084	0.736	0.732	**0.737**	0.731
	CIC-IDS-2018	0.573	**0.642**	**0.642**	0.627	0.757	0.777	**0.780**	0.748
	ToN-IoT	-0.078	-0.072	-0.072	**-0.041**	0.470	0.470	**0.481**	0.447

6.9.2 Performance of ULTRA

Now that we have established the parameter values for the SSTCA$^+$ algorithm, it is time to analyze the performance of ULTRA. First, we discuss the performance of ULTRA with the selected classification models. Second, we show that the UL-

TRA procedure outperforms no embedding and merely an SSTCA$^+$ embedding. Finally, we compare the learning curves of ULTRA with those of ALTRA.

We selected 10,000 instances for source, target, and evaluation datasets to test ULTRA. In addition, we will only label a few instances to test the model's performance in the cold-start setting. We set $q = 5$ and $T = 20$ so that we will eventually label 100 instances. Two additional assumptions we have to make are the number of instances ($\check{m}$,$\check{n}$) selected for the SSTCA$^+$ algorithm and the model f for the update weighing model. We set them both on $1,000$ for the former and selected the random forest model with confidence-based predictions. Figure 6.4 shows the learning curves with one standard deviation confidence bounds of ULTRA using the four different selected ML models when trained on the labeled embedded source and target data created by ULTRA. The results show that the RF and DT work best compared to the results of SVM and NN. Also, with sometimes even five random instances selected, the DD baseline (0.0) is beaten, showing that models can quickly learn from the data. Finally, it can also be observed how quickly the learning curves bend to the limit of what the models can understand.

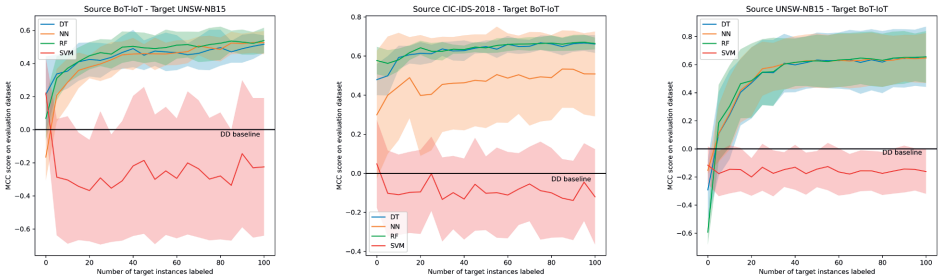


Figure 6.4: Cold-start learning curves using ULTRA with different evaluation classifiers

Now that we have established which evaluation classifier would work in the ID task combined with ULTRA, we still need to show the effect of combining source and target data with and without embedding of ULTRA, or simply SSTCA$^+$. Figures 6.5 show the results of source-target combinations comparing the learning curve of merely performing SSTCA$^+$ or only the labeled target data. The results show that acquiring more target data will improve the performance of the random classifier. For each method, 50 random seeds are used, and we see distributions of the learning curves in the figures. Each plot also has confidence bounds of the learning curves with a standard deviation of 1.0. In these scenarios, the ULTRA or TL improves the models compared to simply using the classifiers on the target data. The results show that merely giving the RF model more data will not necessarily make the model better. It can even introduce noise, making the model

inferior. Embedding the data into another dataspace and even a more sophisticated model, such as ULTRA, can help construct a better classifier.

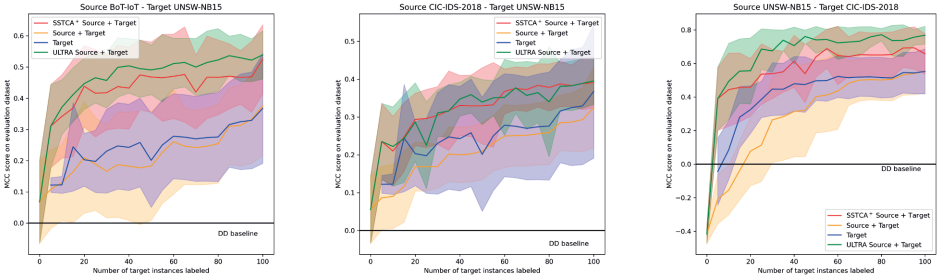


Figure 6.5: Learning curves of an RF trained on different feature spaces

Let us now compare the performance of ALTRA with that of ULTRA. To compute the learning curves of ALTRA, we selected the parameters described in Yuan *et al.* [16] and used an uncertainty random sampling procedure to select instances. The query size and the number of iterations are also selected to be the same (query size $q = 5$ and $T = 20$ iterations). The random forest model is selected as the evaluation model to compare the methods fairly. Figure 6.6 shows the learning curves of both procedures for three different source and target combinations. Each plot also has confidence bounds of the learning curves with a standard deviation of 1.0. The results show that the ALTRA method struggles to even beat the DD baseline, while ULTRA's learning curves show better results. Table 6.5 gives the numeric average results over 50 different random seeds when 100 instances are labeled for both procedures. In bold is highlighted which of the two approaches performs better in each scenario and evaluation metric combination. Looking at the MCC scores, ULTRA beats ALTRA in all source and target dataset combinations. To have a more realistic view of how much the ULTRA model has learned, the DSPI (as described in Section 6.8.1) values are computed, giving a more realistic view of the performance of the random forest classifier when presented with the embedded data.

6.10 Discussion and Conclusion

The integration of TL and AL within the proposed ULTRA framework successfully addresses key challenges in ID, particularly the cold-start problem and DA in dynamic environments. The results from our experiments demonstrate ULTRA's ability to outperform other methods, such as ALTRA and standalone SSTCA($^+$), by embedding source and target data into a shared space and employing an iterative weight update mechanism to prioritize informative instances.

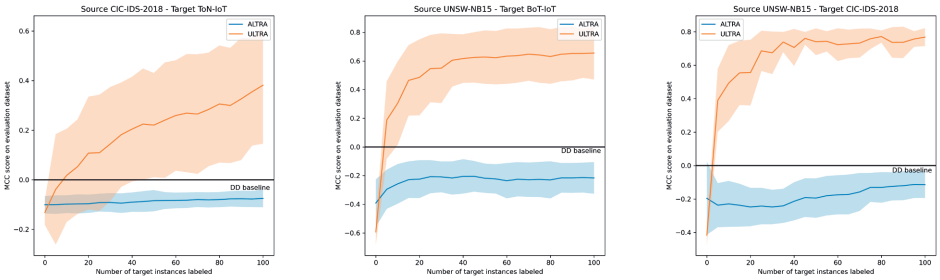


Figure 6.6: Comparison of ALTRA and ULTRA in cold-start scenarios

Table 6.5: Performance comparison of ALTRA and ULTRA when terminated at 100 labeled instances

		MCC		DSPI(MCC)	
Source	**Target**	**ALTRA**	**ULTRA**	**ALTRA**	**ULTRA**
BoT-IoT	CIC-IDS-2018	0.306	**0.553**	0.181	**0.471**
	ToN-IoT	0.132	**0.405**	0.057	**0.288**
	UNSW-NB15	0.330	**0.539**	0.205	**0.451**
CIC-IDS-2018	BoT-IoT	0.655	**0.664**	0.601	**0.612**
	ToN-IoT	-0.075	**0.382**	0.000	**0.305**
	UNSW-NB15	0.052	**0.395**	0.017	**0.273**
ToN-IoT	BoT-IoT	0.177	**0.615**	0.077	**0.570**
	CIC-IDS-2018	0.148	**0.571**	0.110	**0.493**
	UNSW-NB15	0.050	**0.394**	0.034	**0.281**
UNSW-NB15	BoT-IoT	-0.216	**0.655**	0.000	**0.612**
	CIC-IDS-2018	-0.114	**0.769**	0.000	**0.742**
	ToN-IoT	-0.060	**0.478**	0.004	**0.377**

The experimental results highlight several notable findings. First, ULTRA effectively overcomes the cold-start phenomenon by embedding source and target data into a shared feature space using the improved SSTCA^{+} algorithm. This embedding enhances the early-stage performance of classifiers, even with minimal labeled target data. ULTRA surpassed the DD baseline with as few as five labeled instances, demonstrating its ability to improve classifier accuracy even in extremely low-data scenarios rapidly. Additionally, the iterative instance-weighing mechanism within ULTRA mitigates the impact of irrelevant source data, ensuring that transferred knowledge is both relevant and beneficial.

Second, ULTRA's scalability and computational efficiency are evident in its ability to handle large datasets. By incorporating instance-based weighing schemes,

the framework efficiently selects subsets of data for embedding, reducing computational overhead without compromising performance. This is particularly evident in comparisons with traditional feature-based TL methods, which often struggle with scalability.

Third, the framework's robustness across diverse domain shifts is noteworthy. Combining UNSW-NB15 with other datasets consistently demonstrated ULTRA's capacity to adapt to varying network traffic distributions. Including labeled target instances enhanced classifier performance, as seen in improved MCC scores and learning curves.

The analysis of parameters such as λ (local preservation), σ (scaling factor for affinity matrix), and r (neighborhood size) revealed their influence on optimization objectives but limited impact on classifier performance. For instance, while varying λ influenced the eigenvalue distributions within SSTCA$^+$, its effect on the MCC was minimal. Similarly, choosing the median or mean for σ resulted in higher objective values but did not consistently enhance the classifier's accuracy. These insights suggest that ULTRA is relatively robust to parameter variations, simplifying its deployment in real-world scenarios.

ULTRA's ability to achieve significant performance improvements with minimal labeled data has profound implications for practical cybersecurity applications. ULTRA provides a scalable and efficient solution for dynamically evolving network environments by leveraging publicly available labeled datasets and addressing DA challenges. This is particularly valuable when labeling efforts are constrained by time or resources.

While ULTRA has demonstrated substantial advancements, several limitations remain. First, the framework assumes homogeneity between source and target datasets, which may not always hold in real-world applications. Extending ULTRA to handle heterogeneous datasets could broaden its applicability. Second, incorporating uncertainty measures within the AL process could further refine instance selection, potentially improving performance in edge cases. Finally, dynamically optimizing SSTCA$^+$ parameters based on dataset characteristics could enhance adaptability, particularly in scenarios with significant domain shifts.

ULTRA represents a significant advancement in integrating AL and TL for network ID, offering a robust and scalable solution to the cold-start problem and DA challenges. ULTRA paves the way for more efficient and adaptive ML systems in cybersecurity by combining feature- and instance-based TL techniques with active instance selection. Future research should focus on extending the framework's capabilities to heterogeneous datasets and exploring advanced parameter optimization techniques to enhance its performance and applicability.

Bibliography

B

[1] D. Zhou, Z. Yan, Y. Fu, and Z. Yao, "A survey on network data collection", *J. Netw. Comput. Appl.*, volume 116, pages 9–23, Aug. 2018. DOI: 10.1016/j.jnca.2018.05.004.

[2] D. Tovarňák, S. Špaček, and J. Vykopal, "Traffic and log data captured during a cyber defense exercise", *Data Brief*, volume 31, page 105 784, Aug. 2020. DOI: 10.1016/j.dib.2020.105784.

[3] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[4] S. Axelsson, "Intrusion detection systems: A survey and taxonomy", Mar. 2000. [Online]. Available: https://tinyurl.com/2cn476py (visited on 20/12/2023).

[5] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (IDPS)", National Institute of Standards and Technology, Gaithersburg, MD, USA, Tech. Rep. NIST SP 800-94, 2007, pages 1–127. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf (visited on 21/02/2025).

[6] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection", in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, 2010, pages 305–316. DOI: 10.1109/SP.2010.25.

[7] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges", *Comput. Secur.*, volume 28, number 1-2, pages 18–28, Feb. 2009. DOI: 10.1016/j.cose.2008.08.003.

[8] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection", *IEEE Commun. Surv. Tut.*, volume 18, number 2, pages 1153–1176, 2016. DOI: 10.1109/COMST.2015.2494502.

[9] Y. Xin, L. Kong, Z. Liu *et al.*, "Machine learning and deep learning methods for cybersecurity", *IEEE Access*, volume 6, pages 35 365–35 381, 2018. DOI: 10.1109/ACCESS.2018.2836950.

[10] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches", *Trans. Emerg. Telecommun. Technol.*, volume 32, number 1, pages 1–29, Jan. 2021. DOI: 10.1002/ett.4150.

[11] V. Paxson, "Bro: A system for detecting network intruders in real-time", in *Proc. 7th USENIX Secur. Symp.*, volume 7, San Antonio, TX, USA, Jan. 1998, pages 1–22.

[12] M. Hossin and M. Sulaiman, "A review on evaluation metrics for data classification evaluations", *Int. J. Data Mining Knowl. Manage. Process*, volume 5, number 2, pages 1–11, Mar. 2015. DOI: 10.5121/ijdkp.2015.5201.

[13] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning* (Adaptive computation and machine learning), 2nd edition. Cambridge, MA, USA: The MIT Press, 2018.

[14] K. Yang, J. Ren, Y. Zhu, and W. Zhang, "Active learning for wireless IoT intrusion detection", *IEEE Wireless Commun.*, volume 25, number 6, pages 19–25, Dec. 2018. DOI: 10.1109/MWC.2017.1800079.

[15] B. Settles, "Active learning literature survey", University of Wisconsin-Madison, Computer Sciences Technical Report 1648, Jan. 2009, pages 1–44. [Online]. Available: http://digital.library.wisc.edu/1793/60660.

[16] Z. Yuan, X. Chen, Z. Cui, and Y. Mu, "ALTRA: Cross-project software defect prediction via active learning and TrAdaBoost", *IEEE Access*, volume 8, pages 30 037–30 049, 2020. DOI: 10.1109/ACCESS.2020.2972644.

[17] D. Pelleg and A. Moore, "Active learning for anomaly and rare-category detection", in *Proc. Advances Neural Inf. Process. Sys.*, volume 17, Vancouver, BC, Canada, 2004, pages 1–8.

[18] S. J. Pan and Q. Yang, "A survey on transfer learning", *IEEE Trans. Knowl. Data Eng.*, volume 22, number 10, pages 1345–1359, Oct. 2010. DOI: 10.1109/TKDE.2009.191.

[19] Y. Yao and G. Doretto, "Boosting for transfer learning with multiple sources", in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, San Francisco, CA, USA: IEEE, Jun. 2010, pages 1855–1862. DOI: 10.1109/CVPR.2010.5539857.

[20] A. Singla, E. Bertino, and D. Verma, "Overcoming the lack of labeled data: training intrusion detection models using transfer learning", in *IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Washington, DC, USA, Jun. 2019, pages 69–74. DOI: 10.1109/SMARTCOMP.2019.00031.

[21] H. Kheddar, Y. Himeur, S. Al-Maadeed, A. Amira, and F. Bensaali, "Deep transfer learning for automatic speech recognition: Towards better generalization", *Knowl. Based Syst.*, volume 277, pages 1–29, Oct. 2023. DOI: 10.1016/j.knosys.2023.110851.

[22] M. Yuan, H.-T. Lin, and J. Boyd-Graber, "Cold-start active learning through self-supervised language modeling", in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds., Online, Nov. 2020, pages 7935–7948. DOI: 10.18653/v1/2020.emnlp-main.637.

[23] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets", *Comput. Secur.*, volume 86, pages 147–167, Sep. 2019. DOI: 10.1016/j.cose.2019.06.005.

[24] 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 classification problems", *J. Appl. Probability*, volume 62, number 2, pages 475–493, Jun. 2025. DOI: 10.1017/jpr.2024.52.

[25] J. Pries, E. P. van de Bijl, J. G. Klein, S. Bhulai, and R. D. van der Mei, "The optimal input-independent baseline for binary classification: The Dutch Draw", *Statistica Neerlandica*, volume 77, number 4, pages 543–554, May 2023. DOI: 10.1111/stan.12297.

[26] E. P. van de Bijl, J. G. Klein, J. Pries, R. D. van der Mei, and S. Bhulai, "Detecting novel application layer cybervariants using supervised learning", *Int. J. Adv. Secur.*, volume 15, number 3 & 4, pages 75–85, 2022. [Online]. Available: http://www.iariajournals.org/security/.

[27] E. P. van de Bijl, J. G. Klein, J. Pries, S. Bhulai, and R. D. van der Mei, "The Dutch Scaler performance indicator: How much did my model actually learn?" *J. Classification*, pages 1–21, Apr. 2025, early access. DOI: 10.1007/s00357-025-09510-9.

[28] E. P. van de Bijl, S. Bhulai, and R. D. van der Mei, "ULTRA: Utilizing transfer learning to overcome the active learning cold-start phenomenon in network intrusion detection", submitted for publication.

[29] B. E. van Leeuwen, A. W. Gansekoele, J. Pries, E. P. van de Bijl, and J. G. Klein, "Explainable kinship: A broader view on the importance of facial features in kinship recognition", *Int. J. Adv. Life Sci.*, volume 14, pages 89–99, 2022. [Online]. Available: https://personales.upv.es/thinkmind/dl/journals/lifsci/lifsci_v14_n34_2022/lifsci_v14_n34_2022_4.pdf.

[30] J. Pries and E. P. van de Bijl, "The DutchDraw Package", 2023. [Online]. Available: https://github.com/joris-pries/DutchDraw.

[31] E. P. van de Bijl, "NIDS", 2022. [Online]. Available: https://github.com/etiennevandebijl/NIDS.

[32] E. P. van de Bijl, "Dutch Scaler", 2023. [Online]. Available: https://github.com/etiennevandebijl/Dutch-Scaler.

[33] E. P. van de Bijl, "ULTRA", 2025. [Online]. Available: https://github.com/etiennevandebijl/ULTRA.

[34] R. Wirth and J. Hipp, "CRISP-DM: Towards a standard process model for data mining", in *Proc. 4th Int. Conf. Practical Appl. Knowl. Discovery Data Mining*, volume 1, Manchester, UK, Apr. 2000, pages 29–39.

[35] J. H. Min and C. Jeong, "A binary classification method for bankruptcy prediction", *Expert Syst. Appl.*, volume 36, number 3, pages 5256–5263, Apr. 2009. DOI: 10.1016/j.eswa.2008.06.073.

[36] R. Couronné, P. Probst, and A.-L. Boulesteix, "Random forest versus logistic regression: A large-scale benchmark experiment", *BMC Bioinf.*, volume 19, number 1, pages 1–14, Dec. 2018. DOI: 10.1186/s12859-018-2264-5.

[37] S. Wang and C. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification", in *Proc. 50th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, series Short Papers, volume 2, Jeju Island, Korea, 2012, pages 90–94.

[38] R. d. A. Araújo, A. L. O. Oliveira, and S. Meira, "A morphological neural network for binary classification problems", *Eng. Appl. Artif. Intell.*, volume 65, pages 12–28, Oct. 2017. DOI: 10.1016/j.engappai.2017.07.014.

[39] H. Raeisi Shahraki, S. Pourahmad, and N. Zare, "K important neighbors: A novel approach to binary classification in high dimensional data", *BioMed Res. Int.*, volume 2017, pages 1–9, 2017. DOI: 10.1155/2017/7560807.

[40] G. G. Sundarkumar and V. Ravi, "Malware detection by text and data mining", in *Proc. IEEE Int. Conf. Comput. Intell. Comput. Res. (ICCIC)*, Enathi, Tamilnadu, India, Dec. 2013, pages 1–6. DOI: 10.1109/ICCIC.2013.6724229.

[41] G. Sergioli, R. Giuntini, and H. Freytes, "A new quantum approach to binary classification", *PLOS ONE*, volume 14, number 5, A. Lund, Ed., pages 1–14, May 2019. DOI: 10.1371/journal.pone.0216224.

[42] G. Muhammad and M. Melhem, "Pathological voice detection and binary classification using MPEG-7 audio features", *Biomed. Signal Process. Control*, volume 11, pages 1–9, May 2014. DOI: 10.1016/j.bspc.2014.02.001.

[43] O. Koyejo, N. Natarajan, P. Ravikumar, and I. S. Dhillon, "Consistent binary classification with generalized performance metrics", in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., volume 27, Montreal, QC, Canada, 2014, pages 1–9.

[44] Z. C. Lipton, C. Elkan, and B. Naryanaswamy, "Optimal thresholding of classifiers to maximize F1 measure", in *Machine Learning and Knowledge Discovery in Databases*, T. Calders, F. Esposito, E. Hüllermeier, and R. Meo, Eds., volume 8725, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pages 225–239. DOI: 10.1007/978-3-662-44851-9_15.

[45] G. Canbek, S. Sagiroglu, T. Taskaya Temizel, and N. Baykal, "Binary classification performance measures/metrics: A comprehensive visualized roadmap to gain new insights", in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (UBMK)*, Antalya, Turkey, Oct. 2017, pages 821–826. DOI: 10.1109/UBMK.2017.8093539.

[46] J. Balayla, "Prevalence threshold (ϕe) and the geometry of screening curves", *PLOS ONE*, volume 15, number 10, A. D. Hutson, Ed., pages 1–12, Oct. 2020. DOI: 10.1371/journal.pone.0240215.

[47] N. Chinchor, "MUC-4 evaluation metrics", in *Proc. 4th Conf. Message Understanding (MUC4)*, McLean, VA, USA, 1992, pages 22–29. DOI: 10.3115/1072064.1072067.

[48] A. Janosi, W. Steinbrunn, M. Pfisterer, and R. Detrano, "Heart Disease", UCI Machine Learning Repository, 1989. DOI: 10.24432/C52P4X.

[49] F. Pedregosa, G. Varoquaux, A. Gramfort *et al.*, "Scikit-learn: Machine learning in Python", *J. Mach. Learn. Res.*, volume 12, number 85, pages 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html.

[50] O. Russakovsky, J. Deng, H. Su *et al.*, "ImageNet large scale visual recognition challenge", *Int. J. Comput. Vis.*, volume 115, number 3, pages 211–252, Dec. 2015. DOI: 10.1007/s11263-015-0816-y.

[51] A. Paszke, S. Gross, F. Massa *et al.*, "PyTorch: An imperative style, high-performance deep learning library", in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett, Eds., volume 32, Vancouver, Canada, 2019, pages 1–12.

[52] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild", in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pages 3730–3738. DOI: 10.1109/ICCV.2015.425.

[53] W. J. Youden, "Index for rating diagnostic tests", *Cancer*, volume 3, number 1, pages 32–35, 1950. DOI: 10.1002/1097-0142(1950)3:1<32::AID-CNCR2820030106>3.0.CO;2-3.

[54] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation", arXiv:2010.16061 [cs, stat], Oct. 2020.

[55] B. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme", *Biochimica et Biophysica Acta - Protein Struct.*, volume 405, number 2, pages 442–451, Oct. 1975. DOI: 10.1016/0005-2795(75)90109-9.

[56] G. U. Yule, "On the methods of measuring association between two attributes", *J. Roy. Statistical Soc.*, volume 75, number 6, pages 579–652, May 1912. DOI: 10.2307/2340126.

[57] T. O. Kvålseth, "Note on Cohen's Kappa", *Psychol. Rep.*, volume 65, number 1, pages 223–226, Aug. 1989. DOI: 10.2466/pr0.1989.65.1.223.

[58] E. B. Fowlkes and C. L. Mallows, "A method for comparing two hierarchical clusterings", *J. Amer. Statistical Assoc.*, volume 78, number 383, pages 553–569, Sep. 1983. DOI: 10.1080/01621459.1983.10478008.

[59] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images", *Mach. Learn.*, volume 30, number 2/3, pages 195–215, 1998. DOI: 10.1023/A:1007452223027.

[60] W. Palmer, "Note on the accuracy of forecasts concerning the rain problem", *US Weather Bur.*, pages 1–4, 1949.

[61] J. T. Schaefer, "The critical success index as an indicator of warning skill", *Weather Forecasting*, volume 5, number 4, pages 570–575, Dec. 1990. DOI: 10.1175/1520-0434(1990)005<0570:TCSIAA>2.0.CO;2.

[62] B. Pirouz, S. Shaffiee Haghshenas, S. Shaffiee Haghshenas, and P. Piro, "Investigating a serious challenge in the sustainable development process: Analysis of confirmed cases of COVID-19 (new type of coronavirus) through a binary classification using artificial intelligence and regression analysis", *Sustainability*, volume 12, number 6, pages 1–21, Mar. 2020. DOI: 10.3390/su12062427.

[63] L. Li, Y. Yu, S. Bai, Y. Hou, and X. Chen, "An effective two-step intrusion detection approach based on binary classification and k-NN", *IEEE Access*, volume 6, pages 12 060–12 073, 2018. DOI: 10.1109/ACCESS.2017.2787719.

[64] C. Buckley, M. O'Reilly, D. Whelan *et al.*, "Binary classification of running fatigue using a single inertial measurement unit", in *Proc. IEEE 14th Int. Conf. Wearable Implantable Body Sensor Netw. (BSN)*, Eindhoven, The Netherlands, May 2017, pages 197–201. DOI: 10.1109/BSN.2017.7936040.

[65] A. Tharwat, "Classification assessment methods", *Appl. Comput. Inform.*, volume 17, number 1, pages 168–192, Jan. 2021. DOI: 10.1016/j.aci.2018.08.003.

[66] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks", *Inf. Process. Manage.*, volume 45, number 4, pages 427–437, Jul. 2009. DOI: 10.1016/j.ipm.2009.03.002.

[67] J. D. Dixon and B. Mortimer, *Permutation Groups* (Graduate Texts in Mathematics). New York, NY, USA: Springer New York, 1996, volume 163. DOI: 10.1007/978-1-4612-0731-3.

[68] M. Artin, *Algebra*, 2nd edition. Boston, MA, USA: Pearson Education, 2011.

[69] C. Wheelus, E. Bou-Harb, and X. Zhu, "Tackling class imbalance in cyber security datasets", in *Proc. IEEE Int. Conf. Inf. Reuse Integration (IRI)*, Salt Lake City, UT, USA, Jul. 2018, pages 229–232. DOI: 10.1109/IRI.2018.00041.

[70] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: An overview", arXiv:2008.05756 [cs, stat], Aug. 2020.

[71] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review", *J. Netw. Comput. Appl.*, volume 36, number 1, pages 16–24, Jan. 2013. DOI: 10.1016/j.jnca.2012.09.004.

[72] J. Zhao, S. Shetty, and J. W. Pan, "Feature-based transfer learning for network security", in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Bal-

timore, MD, USA, Oct. 2017, pages 17–22. DOI: 10.1109/MILCOM.2017.8170749.

[73] J. Zhao, S. Shetty, J. W. Pan, C. Kamhoua, and K. Kwiat, "Transfer learning for detecting unknown network attacks", *EURASIP J. Inf. Secur.*, volume 2019, number 1, pages 1–13, Dec. 2019. DOI: 10.1186/s13635-019-0084-4.

[74] P. Wu, H. Guo, and R. Buckland, "A transfer learning approach for network intrusion detection", in *Proc. IEEE 4th Int. Conf. Big Data Analytics (ICBDA)*, Suzhou, China, Mar. 2019, pages 281–285. DOI: 10.1109/ICBDA.2019.8713213.

[75] Z. Taghiyarrenani, A. Fanian, E. Mahdavi, A. Mirzaei, and H. Farsi, "Transfer learning based intrusion detection", in *Proc. IEEE 8th Int. Conf. Comput. Knowl. Eng. (ICCKE)*, Mashhad, Iran, Oct. 2018, pages 92–97. DOI: 10.1109/ICCKE.2018.8566601.

[76] M. Masum and H. Shahriar, "TL-NID: Deep neural network with transfer learning for network intrusion detection", in *Proc. IEEE 15th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, London, UK, Dec. 2020, pages 1–7. DOI: 10.23919/ICITST51030.2020.9351317.

[77] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization", in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, Funchal, RAM, Portugal, 2018, pages 108–116. DOI: 10.5220/0006639801080116.

[78] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018)", Canadian Institute for Cybersecurity, 2018. [Online]. Available: https://registry.opendata.aws/cse-cic-ids2018.

[79] A. H. Lashkari, G. Draper Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features", in *Proc. 3rd Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, Porto, Portugal, 2017, pages 253–262. DOI: 10.5220/0006105602530262.

[80] M. Bijone, "A survey on secure network: Intrusion detection & prevention approaches", *Amer. J. Inf. Syst.*, volume 4, number 3, pages 69–88, 2016. DOI: 10.12691/ajis-4-3-2.

[81] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)", in *Proc. IEEE Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Canberra, Australia, Nov. 2015, pages 1–6. DOI: 10.1109/MilCIS.2015.7348942.

[82] 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 *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI)*, Santiago, Chile, Dec. 2018, pages 784–787. DOI: 10.1109/WI.2018.00017.

[83] "Urllib". [Online]. Available: https://docs.python.org/3/library/urllib.html.

[84] G. Canbek, T. Taskaya-Temizel, and S. Sagiroglu, "PToPI: A comprehensive review, analysis, and knowledge representation of binary classification performance measures/metrics", *SN Comput. Sci.*, volume 4, number 1, pages 1–30, Oct. 2022. DOI: 10.1007/s42979-022-01409-1.

[85] G. Canbek, T. Taskaya-Temizel, and S. Sagiroglu, "BenchMetrics: A systematic benchmarking method for binary classification performance metrics", *Neural Comput. Appl.*, volume 33, number 21, pages 14 623–14 650, Nov. 2021. DOI: 10.1007/s00521-021-06103-6.

[86] C. Ferri, J. Hernández-Orallo, and R. Modroiu, "An experimental comparison of performance measures for classification", *Pattern Recognit. Lett.*, volume 30, number 1, pages 27–38, Jan. 2009. DOI: 10.1016/j.patrec.2008.08.010.

[87] D. Brzezinski, J. Stefanowski, R. Susmaga, and I. Szczech, "Visual-based analysis of classification measures and their properties for class imbalanced problems", *Inf. Sci.*, volume 462, pages 242–261, Sep. 2018. DOI: 10.1016/j.ins.2018.06.020.

[88] M. Gösgens, A. Zhiyanov, A. Tikhonov, and L. Prokhorenkova, "Good classification measures and how to find them", in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Type: Conference paper, volume 21, Virtual, 2021, pages 17 136–17 147.

[89] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation", *BMC Genomics*, volume 21, number 1, pages 1–13, Jan. 2020. DOI: 10.1186/s12864-019-6413-7.

[90] R. Delgado and X.-A. Tibau, "Why Cohen's Kappa should be avoided as performance measure in classification", *PLOS ONE*, volume 14, number 9, pages 1–26, Sep. 2019, Publisher: Public Library of Science. DOI: 10.1371/journal.pone.0222916.

[91] A. Luque, A. Carrasco, A. Martín, and A. De Las Heras, "The impact of class imbalance in classification performance metrics based on the binary confusion matrix", *Pattern Recognit.*, volume 91, pages 216–231, Jul. 2019. DOI: 10.1016/j.patcog.2019.02.023.

[92] G. Canbek, T. Temizel-Taskaya, and S. Sagiroglu, "Accuracy barrier (ACCBAR): A novel performance indicator for binary classification", in *Proc. IEEE 15th Int. Conf. Inf. Secur. Cryptography (ISCTURKEY)*, Ankara,

Turkey, Oct. 2022, pages 92–97. DOI: 10.1109/ISCTURKEY56345.2022.9931888.

[93] P. P. Texel, "Measure, metric, and indicator: An object-oriented approach for consistent terminology", in *Proc. IEEE Southeastcon*, Jacksonville, FL, USA, Apr. 2013, pages 1–5. DOI: 10.1109/SECON.2013.6567438.

[94] A. Jain, H. Patel, L. Nagalapatti *et al.*, "Overview and importance of data quality for machine learning tasks", in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Virtual, CA, USA, Aug. 2020, pages 3561–3562. DOI: 10.1145/3394486.3406477.

[95] N. Gupta, S. Mujumdar, H. Patel *et al.*, "Data quality for machine learning tasks", in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining (KDD)*, Virtual, Singapore, Aug. 2021, pages 4040–4041. DOI: 10.1145/3447548.3470817.

[96] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization", *IEEE Trans. Evol. Computat.*, volume 1, number 1, pages 67–82, Apr. 1997. DOI: 10.1109/4235.585893.

[97] L. Kuncheva, C. Whitaker, C. Shipp, and R. Duin, "Limits on the majority vote accuracy in classifier fusion", *Pattern Anal. Appl.*, volume 6, number 1, pages 22–31, Apr. 2003. DOI: 10.1007/s10044-002-0173-7.

[98] V. C. Raykar, S. Yu, L. H. Zhao *et al.*, "Supervised learning from multiple experts: Whom to trust when everyone lies a bit", in *Proc. 26th Annu. Int. Conf. Mach. Learn. (ICML)*, Montreal, QC, Canada, Jun. 2009, pages 889–896. DOI: 10.1145/1553374.1553488.

[99] T. Ishida, I. Yamane, N. Charoenphakdee, G. Niu, and M. Sugiyama, "Is the performance of my deep network too good to be true? A direct approach to estimating the bayes error in binary classification", in *Proc. 11th Int. Conf. Learn. Representations (ICLR)*, Kigali, Rwanda, 2023, pages 1–22.

[100] A. Antos, L. Devroye, and L. Györfi, "Lower bounds for Bayes error estimation", *IEEE Trans. Pattern Anal. Mach. Intell.*, volume 21, number 7, pages 643–645, Jul. 1999. DOI: 10.1109/34.777375.

[101] R. A. Adams and C. Essex, *Calculus: A Complete Course*, 7th edition. Toronto: Pearson Canada, 2010.

[102] G. Canbek, T. Taskaya-Temizel, and S. Sagiroglu, "Binary-Classification Performance Evaluation Reporting Survey Data with the Findings", Mendeley Data, Jul. 2020. DOI: 10.17632/5C442VBJZG.3.

[103] J. A. M. Sidey-Gibbons and C. J. Sidey-Gibbons, "Machine learning in medicine: A practical introduction", *BMC Med. Res. Methodology*,

volume 19, number 1, pages 1–18, Dec. 2019. DOI: 10.1186/s12874-019-0681-4.

[104] W. Wolberg, O. Mangasarian, N. Street, and W. Street, "Breast Cancer Wisconsin (Diagnostic)", UCI Machine Learning Repository, 1993. DOI: 10.24432/C5DW2B.

[105] S. Einy, C. Oz, and Y. D. Navaei, "The anomaly- and signature-based IDS for network security using hybrid inference systems", *Math. Problems Eng.*, volume 2021, A. A. R. Hosseinabadi, Ed., pages 1–10, Mar. 2021. DOI: 10.1155/2021/6639714.

[106] J. Li, W. Wu, and D. Xue, "An intrusion detection method based on active transfer learning", *Intell. Data Anal.*, volume 24, number 2, pages 363–383, Mar. 2020. DOI: 10.3233/IDA-194487.

[107] G. Matasci, M. Volpi, M. Kanevski, L. Bruzzone, and D. Tuia, "Semisupervised transfer component analysis for domain adaptation in remote sensing image classification", *IEEE Trans. Geosci. Remote Sens.*, volume 53, number 7, pages 3550–3564, Jul. 2015. DOI: 10.1109/TGRS.2014.2377785.

[108] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis", *IEEE Trans. Neural Netw.*, volume 22, number 2, pages 199–210, Feb. 2011. DOI: 10.1109/TNN.2010.2091281.

[109] M. Macas, C. Wu, and W. Fuertes, "A survey on deep learning for cybersecurity: Progress, challenges, and opportunities", *Comput. Netw.*, volume 212, pages 1–33, Jul. 2022. DOI: 10.1016/j.comnet.2022.109032.

[110] N. Görnitz, M. Kloft, K. Rieck, and U. Brefeld, "Active learning for network intrusion detection", in *Proc. 2nd ACM Workshop Secur. Artif. Intell. (AISec)*, Chicago, IL, USA, Nov. 2009, pages 47–54. DOI: 10.1145/1654988.1655002.

[111] J. Klein, S. Bhulai, M. Hoogendoorn, and R. D. van der Mei, "Plusmine: Dynamic active learning with semi-supervised learning for automatic classification", in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI-IAT)*, Essendon, VIC, Australia, Dec. 2021, pages 146–153. DOI: 10.1145/3486622.3493948.

[112] J. L. G. Torres, C. A. Catania, and E. Veas, "Active learning approach to label network traffic datasets", *J. Inf. Secur. Appl.*, volume 49, pages 1–13, Dec. 2019. DOI: 10.1016/j.jisa.2019.102388.

[113] D. Kale and Y. Liu, "Accelerating active learning with transfer learning", in *Proc. IEEE 13th Int. Conf. Data Mining (ICDM)*, Dallas, TX, USA, Dec. 2013, pages 1085–1090. DOI: 10.1109/ICDM.2013.160.

[114] D. Kale, M. Ghazvininejad, A. Ramakrishna, J. He, and Y. Liu, "Hierarchical active transfer learning", in *Proc. SIAM Int. Conf. Data Mining (SDM)*, Vancouver, BC, Canada, Jun. 2015, pages 514–522. DOI: 10.1137/1.9781611974010.58.

[115] J. G. Klein, S. Bhulai, M. Hoogendoorn, and R. D. van der Mei, "Jasmine: A new active learning approach to combat cybercrime", *Mach. Learn. Appl.*, volume 9, pages 1–15, Sep. 2022. DOI: 10.1016/j.mlwa.2022.100351.

[116] S. Gou, Y. Wang, L. Jiao, J. Feng, and Y. Yao, "Distributed transfer network learning based intrusion detection", in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Appl. (ISPA)*, Chengdu, Sichuan, China, 2009, pages 511–515. DOI: 10.1109/ISPA.2009.92.

[117] X. Shi, Q. Liu, W. Fan, P. S. Yu, and R. Zhu, "Transfer learning on heterogenous feature spaces via spectral transformation", in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Sydney, Australia, Dec. 2010, pages 1049–1054. DOI: 10.1109/ICDM.2010.65.

[118] C. Wang and S. Mahadevan, "Heterogeneous domain adaptation using manifold alignment", in *Proc. 22nd Int. Joint Conf. Artif. Intell. (IJCAI)*, Barcelona, Spain, 2011, pages 1541–1546.

[119] Z. Peng, W. Zhang, N. Han, X. Fang, P. Kang, and L. Teng, "Active transfer learning", *IEEE Trans. Circuits Syst. Video Technol.*, volume 30, number 4, pages 1022–1036, Apr. 2020. DOI: 10.1109/TCSVT.2019.2900467.

[120] Z. Zhu, X. Zhu, Y. Ye, Y.-F. Guo, and X. Xue, "Transfer active learning", in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, Glasgow Scotland, UK, Oct. 2011, pages 2169–2172. DOI: 10.1145/2063576.2063918.

[121] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning", in *Proc. 24th Int. Conf. Mach. Learn. (ICML)*, series ICML '07, Corvalis, OR, USA, Jun. 2007, pages 193–200. DOI: 10.1145/1273496.1273521.

[122] F. Zhuang, Z. Qi, K. Duan *et al.*, "A comprehensive survey on transfer learning", *Proc. IEEE*, volume 109, number 1, pages 43–76, Jan. 2021. DOI: 10.1109/JPROC.2020.3004555.

[123] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test", *J. Mach. Learn. Res.*, volume 13, number 25, pages 723–773, 2012. [Online]. Available: http://jmlr.org/papers/v13/gretton12a.html.

[124] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring statistical dependence with Hilbert-Schmidt norms", in *Algorithmic Learning*

Theory, D. Hutchison, T. Kanade, J. Kittler *et al.*, Eds., volume 3734, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pages 63–77. DOI: 10.1007/11564089_7.

[125] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples", *J. Mach. Learn. Res.*, volume 7, number 85, pages 2399–2434, 2007. [Online]. Available: http://jmlr.org/papers/v7/belkin06a.html.

[126] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set", in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl. (CISDA)*, Ottawa, ON, Canada, Jul. 2009, pages 1–6. DOI: 10.1109/CISDA.2009.5356528.

[127] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation", in *Proc. 1st Workshop Building Anal. Datasets Gathering Experience Returns Secur. (BADGERS)*, Salzburg, Austria, Apr. 2011, pages 29–36. DOI: 10.1145/1978672.1978676.

[128] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of things for network forensic analytics: Bot-IoT dataset", *Future Gener. Comput. Syst.*, volume 100, pages 779–796, Nov. 2019. DOI: 10.1016/j.future.2019.05.041.

[129] N. Moustafa, M. Ahmed, and S. Ahmed, "Data analytics-enabled intrusion detection: Evaluations of ToN_IoT linux datasets", in *Proc. IEEE 19th Int. Conf. Trust Secur. Privacy Comput. Commun. (TrustCom)*, Guangzhou, China, Dec. 2020, pages 727–735. DOI: 10.1109/TrustCom50675.2020.00100.

[130] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "NetFlow datasets for machine learning-based network intrusion detection systems", in *Big Data Technologies and Applications*, Z. Deze, H. Huang, R. Hou, S. Rho, and N. Chilamkurti, Eds., volume 371, Series Title: Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Cham: Springer International Publishing, 2021, pages 117–135. DOI: 10.1007/978-3-030-72802-1_9.

Summary

In our digital world, we are increasingly dependent on online services. This applies not only to individuals but also to businesses and government institutions. The widespread use of the internet offers many advantages, but it also introduces vulnerabilities to *cyberattacks*, such as phishing emails, viruses, and denial-of-service attacks. Traditional security techniques, such as *signature-based* detection, are struggling to keep up with the ever-evolving and increasingly sophisticated threats. *Cybercriminals* exploit vulnerabilities in software, networks, and human behavior, rendering conventional static defenses less effective. To address these challenges, the use of advanced threat detection techniques has become crucial in modern cybersecurity.

Machine learning (ML) algorithms offer significant potential for *advanced threat detection* in cybersecurity, enabling more effective detection of cyberattacks than traditional methods. ML is a subfield of *artificial intelligence* that allows computers not only to recognize patterns in data, but also to make predictions without being explicitly programmed. In cybersecurity, data sources such as network traffic logs are crucial for identifying potential digital incidents. ML techniques leverage this data to detect attack patterns and help identify malicious activity more quickly. However, applying ML algorithms in cybersecurity brings several challenges. They require high-quality data, careful parameter selection, and thorough evaluation of their effectiveness. Obstacles in cybersecurity data that complicate the use of ML in this domain include: (1) the large imbalance in the data (most connections are normal, while only a small fraction are malicious) and (2) the frequent absence of labels (i.e., whether an instance is benign or malicious). Nevertheless, ongoing developments in ML continue to help overcome these challenges.

Despite the aforementioned challenges, ML techniques remain a powerful tool for detecting cyberattacks. To make these techniques even more reliable, better algorithms are needed that are rigorously tested and whose functioning is explainable. These are essential aspects for real-world deployment. This dissertation examines both the fundamental aspects and the practical application of ML within cybersecurity. It emphasizes that the role of ML is a crucial component of modern cybersecurity. The dissertation is structured into three parts: *baseline methods*, *intrusion detection*, and *learning & breakthroughs*. Each part addresses key challenges in advancing ML-driven cybersecurity.

Baseline Methods

In Chapter 2, we examine the fundamental question of how to evaluate the performance of *binary classification* models. These models aim to predict whether data points belong to one of two categories. For example, whether a connection is malicious or not. However, drawing meaningful conclusions from an *evaluation metric* score, such as *accuracy*, can be difficult. What does it really mean when a *classifier* achieves an *accuracy* of 0.8? The structure of the data can introduce *bias*, making a model's performance appear better or worse than it truly is. This chapter introduces the *Dutch Draw* (DD), a universal *baseline* method for assessing the performance of binary classifiers. This method establishes a *baseline* that any meaningful model should at least outperform. The DD is *general*, *simple*, and *informative*, ensuring that performance scores are correctly contextualized and can be meaningfully interpreted. It provides a clear and explainable theoretical reference point for existing *evaluation metrics*, and should be seen as the 'golden standard' for model comparison. If a model does not outperform this baseline, this should be taken as a warning. The DD method simplifies both model development and evaluation, and offers clearer insight into the performance of *binary classification* models.

In Chapter 3, we broaden the scope of model evaluation by examining all *input-independent binary classification baselines*, which do not rely on any data features. We compare these classifiers and determine what constitutes an 'optimal' baseline. Most standard *evaluation metrics* in *binary classification* are *order-invariant*, meaning that they are unaffected by the order of the data points. We define optimality using the concept of *average-permutation-optimality*, which evaluates the expected performance of a classifier under the assumption that both data and predictions are randomly permuted. The main result, which we formally prove, is that the DD baseline outperforms all other possible *input-independent classifiers* for any *order-invariant evaluation metric*, thereby establishing the DD as the 'golden standard' for evaluating binary classifiers.

Intrusion Detection

In Chapter 4, we investigate the ability of binary classification models to detect novel variants of known cyberattacks, specifically focusing on (distributed) denial-of-service and web attacks targeting the application layer (HTTP). This chapter illustrates how datasets are constructed and prepared for ML use by combining features from the network, transport, and application layers. Three experimental setups are tested: (1) detecting known attacks, (2) evaluating detection performance on previously unseen variants, and (3) analyzing the effect of training with multiple attack types. The results show that classifiers are generally effective at detecting known attacks. Some novel variants can be detected when classifiers are trained on similar attacks. However, detection performance is not always symmetric: if a classifier can detect one attack when trained on another, the reverse is not necessarily true. Moreover, increasing the number of attack types in the training set does not always improve detection performance, suggesting that a well-chosen subset of known attacks may be more effective. The DD baseline is also used for benchmarking the classifiers in this study, highlighting that ML can detect cyberattacks at performance levels comparable to signature-based detection methods, while also being capable of detecting novel variants under the right conditions.

Learning and Breakthroughs

In Chapter 5, we shift our focus to quantifying how much a model has actually learned. In pursuit of this, we introduce the *Dutch Scaler* (DS), a novel *performance indicator* that captures this aspect. The *Dutch Scaler performance indicator* (DSPI) contextualizes empirical metric scores using two reference points: the DD baseline, which represents a classifier that does not learn from data, and the *Dutch Oracle* (DO), which approximates an 'optimal' classifier. By mapping evaluation scores onto a *uniform reference framework*, the DSPI provides a more meaningful interpretation of evaluation metrics. The study demonstrates how the DSPI enables comparison of classifiers across multiple metrics. The DS framework is adaptable and has potential extensions to multi-class and regression problems, making it a valuable addition to existing evaluation methodologies in ML.

Classifiers rely on labeled data to learn. While large volumes of data are often available, labels are frequently scarce. This leads to the so-called *cold-start phenomenon*, which can occur when a classifier needs to be deployed in a new network environment. Fortunately, labeled datasets from accessible *source* domains can be leveraged to improve the performance of classifiers that only have access to unlabeled *target* data. However, such source data are often not directly usable for traditional ML models due to distributional differences between source

and target. In Chapter 6, we introduce ULTRA, a framework that combines *active* and *transfer learning* to address this challenge in *network intrusion detection*. ULTRA combines *instance-based* and *feature-based* transfer learning techniques to project source and target data into a shared space, while continuously refining instance selection for this transformation. The results highlight ULTRA's performance under cold-start conditions. The framework demonstrates scalability and efficiency, particularly in scenarios with limited labeled data. Moreover, ULTRA exhibits robustness to domain shifts, making it well-suited for real-world applications in cybersecurity. Nonetheless, limitations remain. In particular, ULTRA assumes that source and target data share an identical *feature space*. Future work should focus on extending ULTRA to settings with differing *feature spaces* and on dynamically optimizing its parameters to further improve adaptability.

Collectively, these chapters present both fundamental research and practical applications of ML in cybersecurity. The aim of this dissertation is to support data scientists and cybersecurity experts by providing valuable tools for developing and evaluating more effective models and methods.

Samenvatting

In onze digitale wereld zijn we steeds meer afhankelijk van online diensten. Dit geldt voor individuen, maar ook voor bedrijven en overheidsinstellingen. Het grootschalig gebruik van het internet heeft veel voordelen, maar het introduceert ook kwetsbaarheid voor *cyberaanvallen* zoals phishing mails, virussen en denial-of-service aanvallen. Traditionele beveiligingstechnieken, zoals bijvoorbeeld *signature-based* detectie, kunnen de steeds verder ontwikkelde en meer geavanceerdere bedreigingen helaas niet voldoende bijbenen. *Cybercriminelen* maken namelijk misbruik van kwetsbaarheid in software, netwerken en menselijk gedrag, waardoor de gebruikelijke statische verdedigings technieken niet langer altijd even effectief zijn. Om deze uitdagingen het hoofd te bieden is het gebruik van *advanced threat* detectietechnieken onmisbaar geworden binnen moderne *cybersecurity*.

Machine learning (ML)-algoritmes bieden veel potentie voor *advanced threat detection* in cybersecurity, waardoor cyberaanvallen effectiever gedetecteerd kunnen worden dan met de traditionele methodieken. ML is een subset gebied binnen *kunstmatige intelligentie* en maakt het mogelijk om met computers patronen te herkennen in data, maar ook om voorspellingen te doen zonder dat dit expliciet wordt aangestuurd. In cybersecurity zijn databronnen, zoals netwerkverkeer, cruciaal voor het detecteren van potentiële digitale incidenten. De technieken in ML maken gebruik van deze data om patronen te herkennen van mogelijke aanvallen voor het sneller opmerken van malicieuze activiteiten. Het toepassen van ML algoritmes in cybersecurity brengt echter verschillende uitdagingen met zich mee. Het toepassen ervan vereist data van hoge kwaliteit, de parameters moeten zorgvuldig gekozen worden en ze moeten grondig getest worden op effectiviteit. Obstakels in cybersecurity data die het gebruik van ML in dit veld lastiger maken zijn echter (1) de grote disbalans in de data - veel connecties zijn normaal, terwijl

slechts een klein deel kwaadaardig is - en (2) het frequente ontbreken van labels (normaal of malicieus). Toch maken de ontwikkelingen binnen ML het mogelijk om deze uitdagingen te tackelen.

ML-technieken zijn ondanks bovenstaande een krachtig hulpmiddel voor het detecteren van cyberaanvallen. Om ML-technieken nog betrouwbaarder te maken zijn er betere algoritmen nodig die grondig getoetst moeten worden en waarvan de werking uitlegbaar is. Dit zijn noodzakelijke aspecten voor de toepassing in de praktijk. In dit proefschrift is zowel de fundamentele kant alswel de praktische toepassing van ML binnen cybersecurity onderzocht. Benadrukt wordt dat de rol van ML een essentieel onderdeel is binnen de moderne cybersecurity. Deze dissertatie is opgedeeld in drie delen: *baseline methodieken*, *intrusiedetectie* en *leren & doorbraken*. Elk deel behandelt belangrijke uitdagingen in het bevorderen van ML-gedreven cybersecurity.

Baseline Methodieken

In Hoofdstuk 2 wordt fundamentaal onderzocht hoe de prestaties van *binary classificatie modellen* geëvalueerd kunnen worden. Deze modellen hebben als doel te voorspellen of datapunten tot een van twee categorieën behoren. Bijvoorbeeld, is een connectie kwaadaardig of niet? Het trekken van betekenisvolle conclusies uit een *evaluation metric* score, zoals *accuracy*, kan echter lastig zijn. Wat betekent het bijvoorbeeld wanneer een *classifier* een *accuracy* van 0,8 behaalt? De structuur van de data kan *bias* introduceren, waardoor de prestatie van een model beter of slechter oogt dan het daadwerkelijk is. In dit hoofdstuk wordt de *Dutch Draw* (DD) geïntroduceerd, een universele baseline techniek, om de prestatie van binary classifiers te kunnen beoordelen. Deze methodiek stelt een *baseline* vast waar andere modellen tenminste superieur aan moeten zijn om van betekenis te zijn. De DD is *generiek*, *simpel* en *informatief* en zorgt ervoor dat prestatiescores correct worden gecontextualiseerd, waardoor een juiste beoordeling mogelijk is. Het biedt een duidelijk en verklaarbaar theoretisch referentiepunt voor bestaande *evaluation metrics* en zou de 'gouden standaard' moeten zijn voor het vergelijken van modellen. Als een model namelijk niet beter presteert dan deze baseline, dan zal dit als een waarschuwing moeten worden beschouwd. De DD-methode vereenvoudigt modelontwikkeling en evaluatie en biedt een duidelijker inzicht in de prestaties van *binary classificatie modellen*.

In Hoofdstuk 3 breiden we de scope van modelevaluatie uit door alle *input-onafhankelijke binary classificatie baselines* te onderzoeken. Deze classifiers maken geen gebruik van enige *features* uit data. We vergelijken dit soort *classifiers* en stellen vast wat een 'optimale' baseline is. De meeste standaard *evaluation metrics* in *binary classification* zijn *order-invariant*, wat betekent dat ze niet worden

beïnvloed door de volgorde van de datapunten. We definiëren het begrip optimaliteit middels het concept van *average-permutation-optimality*. Hierbij wordt gekeken naar de verwachte prestatie van een *classifier* onder de aanname dat data en voorspellingen willekeurig gepermuteerd zijn. Het belangrijkste resultaat in dit hoofdstuk is dat we formeel bewijzen dat de DD-baseline beter presteert dan alle andere mogelijke *input-onafhankelijke classifiers* voor elke *order-invariant evaluation metric*. Hierdoor is de DD de 'gouden standaard' voor het evalueren van *binary classifiers*.

Intrusiedetectie

In Hoofdstuk 4 onderzoeken we het vermogen van binaire classificatie modellen om nieuwe varianten van bekende cyberaanvallen te detecteren. Hierbij ligt de focus op (gedistribueerde) denial-of-service én web aanvallen, die gericht zijn op de applicatielaag HTTP. Dit hoofdstuk laat zien hoe datasets worden geconstrueerd, die gereed zijn voor het gebruik van ML. Dit wordt gedaan door *features* van netwerk-, transport- en applicatie-laag samen te voegen. De ML classifiers zijn in drie experimentele configuraties getoetst: (1) het detecteren van bekende aanvallen, (2) het evalueren van detectieprestaties op niet eerder geziene varianten en (3) het analyseren van het effect van het gebruik maken van meerdere aanvalstypes. De resultaten tonen aan dat classifiers over het algemeen goed in staat zijn bekende aanvallen te detecteren. Sommige nieuwe varianten kunnen gedetecteerd worden wanneer classifiers zijn getraind op vergelijkbare aanvallen. De detectieprestaties zijn echter niet altijd symmetrisch: als een classifier in staat is om een aanval te detecteren, wanneer het geleerd heeft om een andere aanval te detecteren, dan is het omgekeerde niet noodzakelijkerwijs ook zo. Bovendien verhoogt het aantal aanvalstypen in de *training* niet altijd de detectie-prestaties, wat suggereert dat een goed gekozen subset van bekende aanvallen effectiever kan zijn. De DD-baseline wordt ook gebruikt voor het *benchmarken* van *classifiers* in deze studie. De resultaten benadrukken dat ML cyberaanvallen kan detecteren, waarbij het prestatieniveau vergelijkbaar is met *signature-based* detectie methodieken. Echter, ML kan ook nieuwe varianten detecteren onder de juiste omstandigheden.

Leren en Doorbraken

In bovenstaande is de uitdaging besproken om te bepalen of een classifier beter presteert dan een baseline. In Hoofdstuk 5 verschuiven we de focus naar het kwantificeren van hoeveel een model daadwerkelijk heeft geleerd. Ter bevordering hiervan introduceren we de *Dutch Scaler* (DS), een nieuw *performance indicator* die dit vastlegt. De *Dutch Scaler performance indicator* (DSPI) contextualiseert empirische *metric* scores met behulp van twee referentiepunten: de DD-baseline, die een classifier vertegenwoordigt die niet leert van data, en de *Dutch Oracle* (DO),

die de benadering van een 'optimale' classifier geeft. De DSPI biedt een nog betekenisvollere interpretatie van *evaluation metrics* door betreffende scores af te beelden in een *uniform referentiekader*. De studie toont aan hoe de DSPI classifiers kan vergelijken over meerdere *metrics*. Het DS framework is aanpasbaar, met potentiële uitbreidingen naar multi-klasse én regressie problemen, waardoor het een waardevolle aanvulling is op de evaluatiemethodologieën binnen ML.

Classifiers hebben gelabelde data nodig om te leren. Hoewel grote hoeveelheden gegevens vaak beschikbaar zijn, zijn de labels dat vaak niet. Dit wordt ook wel het zogenaamde *cold-start fenomeen* genoemd en kan optreden als een classifier gebouwd moet worden in een nieuw netwerk. Gelukkig kunnen andere gelabelde toegankelijke datasets (*sources*) worden gebruikt om de prestaties van classifiers te verbeteren die alleen niet-gelabelde dataset (*target*) tot hun beschikking hebben. Het is alleen vaak niet het geval dat deze source data direct bruikbaar zijn voor traditionele ML classifiers door verdelingsverschillen tussen source en target. In Hoofdstuk 6 introduceren we ULTRA, een framework dat *active* en *transfer learning* combineert om deze uitdaging in *network intrusiedetectie* aan te pakken. ULTRA combineert *instance-based* en *feature-based* transfer learning technieken om source- en target data naar een gedeelde ruimte te transformeren, terwijl het continu de instance-selectie verbetert voor het maken van deze transformatie. De resultaten benadrukken de prestaties van ULTRA in de cold-start situatie. Het framework toont schaalbaarheid en efficiëntie aan, met name in scenario's met beperkt gelabelde data. ULTRA is robuust tegen mogelijke *domain shifts* en is hierdoor toepasbaar binnen het cybersecurity domein. Er blijven echter beperkingen van ULTRA, zoals de aanname van identieke *feature spaces* tussen source- en target data. Toekomstig onderzoek zou zich moeten richten op het aanpassen van ULTRA voor datasets met verschillende *feature spaces* en het dynamisch optimaliseren van de parameters.

Deze hoofdstukken presenteren samen zowel fundamenteel onderzoek als de praktische toepassingen van ML in cybersecurity. Het doel van deze dissertatie is om datawetenschappers en cybersecurity-experts te ondersteunen met waardevolle hulpmiddelen voor het bouwen en evalueren van effectievere modellen en methoden.

Dankwoord

Dit proefschrift is het resultaat van jaren onderzoek, maar had nooit tot stand kunnen komen zonder de steun, samenwerking en ideeën van velen. In deze sectie wil ik de mensen bedanken die op verschillende manieren een waardevolle rol hebben gespeeld tijdens mijn promotietraject.

Allereerst wil ik mijn promotoren, *Rob van der Mei* en *Sandjai Bhulai*, bedanken. Jullie begeleiding, vertrouwen en enthousiasme hebben mijn promotietraject niet alleen leerzaam, maar ook erg plezierig gemaakt. Onze wekelijkse overleggen waren gezellig en inspirerend. Jullie hielpen mij enorm bij het sturen van het onderzoek in de juiste richting. Rob, jij bent geweldig in het uitleggen van de waarde van onderzoek, op een manier die iedereen begrijpt en waar mensen enthousiast van worden. Daar heb ik veel van geleerd. Naast onze fijne samenwerking binnen het promotieonderzoek was het ook leuk om samen het vak *Performance of Networked Systems* te doceren, de actuele politieke en voetbalontwikkelingen te bespreken en soms een stukje samen te fietsen wanneer we elkaar tegenkwamen op weg naar het CWI. Dat we toevallig in dezelfde Uilenstede-eenheid hebben gewoond blijft een grappig feitje. Sandjai, jouw optimisme en waardevolle inzichten gaven mij het vertrouwen om dit traject goed af te ronden. Ik heb altijd bewonderd hoe jij in podcasts/colleges/video's op een geweldige manier kon laten zien wat je allemaal met wiskunde kunt doen. Ik waardeer het enorm hoe jullie mij de vrijheid gaven om mijn eigen pad te volgen, maar tegelijkertijd wisten bij te sturen met waardevolle feedback. Het is een eer om onder jullie begeleiding te mogen promoveren.

Tevens wil ik de promotiecommissie bedanken voor het lezen en beoordelen van mijn proefschrift. Hun feedback en inzichten worden erg gewaardeerd. Ik kijk uit naar de discussie en het uitwisselen van ideeën tijdens de verdediging.

Dit proefschrift is tevens het resultaat van een samenwerkingsproject tussen het *Centrum Wiskunde & Informatica* (CWI) en het *Ministerie van Binnenlandse Zaken en Koninkrijksrelaties* (MinBZK). Mijn dank gaat uit naar iedereen binnen MinBZK die heeft bijgedragen aan dit traject. Jullie inzet en overtuiging in het doen van wetenschappelijk onderzoek hebben deze samenwerking mogelijk gemaakt. Hierdoor kon ik mijn onderzoek in de richting ontwikkelen die ik zelf wilde, en daar ben ik erg dankbaar voor.

Het startschot van mijn tijd in dit project begon met mijn masterscriptie onder begeleiding van *Jan* en *Joris*. In 2018 vertelde Jan over zijn onderzoek in cybersecurity. Dit motiveerde mij om in dit vakgebied een masterscriptie te willen schrijven. Jan en Joris werden mijn dagelijkse supervisors tijdens mijn scriptie. Toen ik later als promovendus verder ging in het project, groeiden we als collega's naar elkaar toe en ontstond een geweldige samenwerking. Dat is ook terug te zien in de artikelen die we samen hebben geschreven. We hebben talloze interessante gesprekken gevoerd, veel van elkaar geleerd en een waardevolle en speciale vriendschap opgebouwd. Jullie brachten zoveel gezelligheid mee, waardoor het een plezier was om naar kantoor te gaan. Zelfs nadat jullie het CWI hadden verlaten, spraken we elkaar nog wekelijks online, en tot op de dag van vandaag sluiten we onze Zoom-meetings af met onze traditionele groet: *"Wetenschap!"*. Jan, de *"Wie is de Mol?"* deskundige, muziekliefhebber en verbinder. Jouw eerlijkheid en blik op zaken hebben mij geholpen om scherp en realistisch naar onderzoek/de wereld te kijken. Jouw toegankelijkheid en scherpe inzichten maakten elk gesprek (tijdens het fietsen) waardevol en gaven altijd nieuwe perspectieven. Joris, de man van de gouden ideeën, powermoves en gadgets. Jij hebt mij geïnspireerd om soms kritisch te kijken naar algemeen geaccepteerde conventies, zoals elke wetenschapper zou moeten doen. Jouw creatieve ideeën hebben mij vaak aan het denken gezet en het onderzoek naar een hoger niveau getild. Ik ben vereerd dat jullie naast mij staan tijdens de verdediging als mijn paranimfen.

Het project werd halverwege mijn promotietraject versterkt door (padelmaatjes) *Arwin* en *Britt*. Beiden organisatorische krachten die ook een grote rol hebben gespeeld in de gezelligheid en de groepsdynamiek. Samen met Britt bezocht ik mijn eerste internationale conferentie, een bijzondere en memorabele mijlpaal, en samen met Arwin organiseerden we verschillende leuke uitjes en borrels. Onze koffiemomenten bij Doppio waren altijd waardevolle momenten om de ontwikkelingen binnen het project te bespreken. Ik wens jullie, evenals de nieuwste aanwinsten van het project, *Simon* en *Moos*, veel succes en wijsheid in (het afronden van) jullie promotietrajecten.

Het grootste deel van mijn promotietijd heb ik doorgebracht bij het CWI, binnen de stochastiek groep onder leiding van *Bert Zwart*. Het was een mooie ervaring om er te mogen werken. De gezellige lunchmomenten met *Marie-Colette* en *Bart* braken de dag op een prettige manier. Later in de middag een potje tafeltennis of volleybal was altijd een welkome afwisseling. En natuurlijk de gezellige borrels bij de polder om de werkweek af te sluiten.

In kamer M326 van het CWI is hardzwoegend het grootste deel van mijn proefschrift geschreven. In deze ruimte zijn ook voor mij hele mooie vriendschappen ontstaan met *Rebekka* en *Salim*. Rebekka, sfeermaker van de groep, onze gesprekken begonnen (online) tijdens de covidperiode en zijn eigenlijk nooit meer gestopt. Je bood altijd nieuwe perspectieven, een luisterend oor en een flinke dosis gezelligheid, waardoor zelfs de zwaarste dagen lichter voelden. Salim, jouw steun betekende ontzettend veel. Van de iconische tekeningen op ons whiteboard tot de prachtige cover van mijn proefschrift; jouw creativiteit maakt een blijvende indruk. Maar bovenal was je een fijne vriend, iets wat ik enorm heb gewaardeerd.

Collega's van de CWI stochastiekgroep die ik daarnaast graag wil bedanken zijn: padelmaatjes *Berend* en *Jasper*, voor jullie speelse humor, motiverende woorden en complimenten; *Ruurd* en *Hilde*, voor de altijd boeiende positieve energie die jullie meenamen; *Robin*, *Elisabeth* en *Guus*, voor jullie gezelligheid tijdens de borrels; de Coconut-boys *Jesse* en *Joris*, met wie ik een onvergetelijke reis naar Indianapolis (USA) maakte voor een conferentie; de Dolce Vita-groep, *Louisa*, *Joost*, *Tim* en *Katja*, die voor veel gezelligheid op de werkvloer zorgden; en natuurlijk *Martijn* en *Ismail* voor de Brabantse/Limburgse gezelligheid. A special thank you to *Eda*, *Barbara*, and *María*, with whom I have had many enjoyable conversations during their master's thesis time.

Op de VU heb ik elke donderdag doorgebracht als onderdeel van de A&O-groep, onder leiding van *Ger Koole*. Ik ben Ger dankbaar voor het organiseren van het wekelijkse A&O-seminar, waar ik mijn academische kennis heb kunnen verbreden. Drie speciale mensen die ik in mijn tijd aan de VU heb leren kennen zijn *Yura*, *Anni* en *Rik*. Tijdens de covidperiode heb ik veel samengewerkt met Yura (en *Emiel*), waarin we overdag hard hebben gewerkt en in de avond veel Mario Kart hebben gespeeld. Onze vriendschap betekent veel voor mij. Anni, my Greek friend and "go-to person" for concerts, we went to so many that I've almost lost count. I truly admire your joy for life. Ik ken Rik al sinds het begin van mijn masterthesis. Zijn doorzettingsvermogen en oprechtheid zijn eigenschappen die ik enorm waardeer. Inmiddels genieten we vaak samen van Ajax-wedstrijden in Café Kuijper.

Tevens wil ik *Mark Hoogendoorn* van de VU QDA groep bedanken voor zijn waardevolle bijdrage aan het Dutch Draw-project. Ik ben ook dankbaar voor de leuke tijd met de anderen VU mede-promovendi/collega's die ik elke donderdag zag: *René*, *Joost*, *Robin*, *Leon*, *Yoram*, *Ritsaart*, *Witek*, *Jeroen* en nog vele andere. Daarnaast heb ik tijdens de jaarlijkse A&O hikingtrip velen beter leren kennen, namelijk: *Erica*, *Jeroen*, *Yasmin*, *Mathijs*, *Jesper*, *Laith*, *Jaap*, *Corné*, *Alessandro* en *Siqiao*. Ik kijk met plezier terug op deze reizen met jullie.

Naast collega's wil ik ook speciale vrienden van daarbuiten bedanken. Allereerst wil ik mijn dierbaarste vrienden *Ruben* en *Ronald* noemen. We zijn al meer dan een decennium vrienden, en onze nostalgische avonden in het dorp waren altijd een fijne afleiding tijdens mijn promotieonderzoek. Onze vriendschap betekent voor mij heel veel. Daarnaast wil ik mijn studievrienden *Nadine*, *Esmee*, *Olav* en *Niels* bedanken voor de vele gezellige weekendjes weg die we samen hebben beleefd. Met ieder van jullie heb ik een speciale band die voor mij enorm waardevol is. Een andere studiegenoot die ik wil bedanken is *Romy*. Al jaren koken we voor elkaar, ondernemen we leuke activiteiten, en dankzij jou heb ik een geweldige vriendengroep leren kennen, waar ik nu ook onderdeel van ben. Ik ben dankbaar voor onze vriendschap. Twee oud-huisgenootjes wil ik ook nog even in de spotlight zetten: *Anouk* en *Hilly*. Samen thuis chillen en bijkletsen was altijd een fijne ontsnapping. Mijn band met Hilly was daarbij extra bijzonder. Van diepgaande gesprekken tot avondjes uitgaan, het is altijd gezellig en vertrouwd.

Tot slot wil ik mijn familie en dierbaren bedanken voor hun steun, waardevolle feedback en de vele fijne momenten die we samen hebben gedeeld. In het bijzonder wil ik mijn ouders *Ed* en *Cora*, zusje *Francine* en vriendin *Vera* noemen. Jullie onvoorwaardelijke steun, liefde en vertrouwen hebben mij door dit traject heen geholpen. Dankzij jullie is *de volgende academische stap gezet*. Dank jullie wel daarvoor.

About the Author

Etienne Pieter van de Bijl was born on May 7, 1994, in Amsterdam, the Netherlands. He grew up in Venhuizen, a small village in North Holland. In 2013, he completed his *vwo* education at RSG Enkhuizen before pursuing a BSc in Business Analytics at Vrije Universiteit (VU) Amsterdam. He graduated in 2017 with his bachelor's thesis titled *Predicting daily payment transactions at ING*. To further deepen his expertise, he pursued an MSc in Business Analytics at VU, specializing in the Computational Intelligence track.

As part of his master's degree, Etienne undertook an internship at the *Dutch Ministry of the Interior and Kingdom Relations* (MinBZK) and *Centrum Wiskunde & Informatica* (CWI) in Amsterdam, under the supervision of Jan Klein and Joris Pries. In his master's thesis, *Towards graph-based intrusion detection in cybersecurity*, he developed a graph-based intrusion detection system that utilized anomaly detection techniques to identify malicious network changes, enhancing the effectiveness of threat detection.

In 2019, Etienne started his PhD research at CWI and MinBZK. This dissertation is the result of that research, which will be defended on October 2, 2025. During his PhD, Etienne assisted in teaching for the course Performance of Networked Systems together with Rob van der Mei. Etienne also presented his work at several international conferences.