Fundamentals of Data Science

A New Outlook

ISBN: 978-94-6473-127-9

Ministerie van Binnenlandse Zaken en
Koninkrijksrelaties

Typeset by: LaTeX.
Printed by: Ipskamp Printing

VRIJE UNIVERSITEIT

Fundamentals of Data Science
A New Outlook

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor of Philosophy aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. J.J.G. Geurts,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de Faculteit der Bètawetenschappen
op maandag 26 juni 2023 om 11.45 uur
in een bijeenkomst van de universiteit,
De Boelelaan 1105

door

Joris Pries

geboren te Houston, Verenigde Staten

promotoren:	prof.dr. R.D. van der Mei
	prof.dr. S. Bhulai
promotiecommissie:	prof.dr. M. Hoogendoorn
	dr. E.J. Bekkers
	dr. J. Berkhout
	prof.dr.ir. G. Jongbloed
	prof.dr. F.M. Spieksma

Aan mijn ouders

Door hun zorg werd het pad
voor mijn academische studie
geëffend.

Voor Annemieke en Bas

Voorwoord

Het voorwoord. De openingstitels van mijn proefschrift. Tegenwoordig is het niet meer gebruikelijk om een film te beginnen met een lange lijst van acteurs en crewleden, omdat onze aandachtsspanne simpelweg te kort is. We willen namelijk meteen actie en sensatie. Vandaar dat in nieuwe films zo'n opsomming vaak te zien is aan het einde van een film. Echter, heb ik té vaak gezien dat mensen opstonden voordat de aftiteling voorbij was. Daarom wil ik tóch het geduld van de lezer hier op de proef stellen, omdat dit proefschrift niet tot stand was gekomen zonder de hulp van anderen. Zij verdienen de aandacht en waardering.

Allereerst wil ik de schijnwerpers richten op mijn twee promotoren: *Rob van der Mei* en *Sandjai Bhulai*. Vanaf de eerste dag voelde ik mij bij jullie op mijn gemak. Jullie gaven mij veel vrijheid en spraken vertrouwen uit, iets dat ik erg waardeerde. Rob, jij kan als geen ander de waarde van ons onderzoek overbrengen. Sandjai, jij hebt altijd razendsnel door wat wiskundig het probleem is. Ik wil jullie bedanken voor jullie begeleiding, humor, en enthousiasme.

Dit proefschrift is voortgekomen uit een gezamenlijk project van het *Centrum Wiskunde & Informatica (CWI)* en het *Ministerie van Binnenlandse Zaken en Koninkrijksrelaties (MinBZK)*. Ik wil alle betrokkenen bij MinBZK bedanken voor hun samenwerking en inzet. Mede hierdoor zal ik deze tijd nooit vergeten. Dit unieke project kon alleen tot stand komen dankzij

jullie doorzettingsvermogen, expertise, en overtuiging dat wetenschappelijk onderzoek doen waardevol is.

Een belangrijk onderdeel van het behalen van het doctoraat is de *promotiecommissie*, wat te vergelijken valt met een groep filmcritici. Indirect zorgen zij ervoor dat je het proefschrift zo goed mogelijk wilt maken. Ik wil hen bedanken voor de tijd en energie die zij hebben gestoken in het lezen en beoordelen van dit proefschrift. Dit wordt erg gewaardeerd. Ook ben ik benieuwd naar de discussiepunten die zij zullen opbrengen tijdens de verdediging.

Twee belangrijke hoofdrolspelers van mijn PhD zijn projectgenoten *Jan Klein* en *Etienne van de Bijl*, waar ik makkelijk een spin-off tv-serie over zou kunnen schrijven. Zij hebben beiden veel betekend voor mij tijdens dit traject. Jullie zijn een onuitputbare bron van afleiding, humor, en gouden ideeën. Een van die ideeën ontstond naar aanleiding van Jan's muntje, waardoor we samen mooi onderzoek hebben gedaan. De Engelse benaming voor een proefschrift 'dissertation' komt van een Latijns woord voor discussie. Iets dat wij veel gedaan hebben de afgelopen vier jaar. Aan het woord 'discussie' zit misschien een negatieve connotatie, maar zo waren onze discussies niet. We konden het eindeloos hebben over wiskundige, maar ook totaal niet-wiskundige onderwerpen. Als een groot Wie is de Mol fan vond ik het heerlijk om met een nog grotere fan (Jan) hierover te sparren. Etienne kwam ook vaak bij ons de kamer binnenlopen voor een wiskundig probleem of om gezellig te praten. Ik denk dat juist dit soort ontspanningen, uiteindelijk mij de ingevingen hebben gegeven voor mijn onderzoek. Ik wil Jan nog prijzen voor zijn toegankelijkheid en oprechtheid, wat mij overhaalde om aan dit project te beginnen. Etienne wil ik roemen voor zijn sociale en organisatorische vaardigheden. Als 'de man van het volk' zorgde jij voor sfeer in de groep door koffiemomenten, feestjes, en groepsuitjes te organiseren. Zo zou ik nog wel even door kunnen gaan, maar er zijn nog meer mensen die de aandacht verdienen voordat de lezer afhaakt.

In de loop van de tijd werd de cast van ons project uitgebreid met *Arwin Gansekoele* en *Britt van Leeuwen*. Twee waardevolle toevoegingen, die het project in de toekomst verder zullen brengen. Britt is organisatorisch sterk, en Arwin (uit Almere) wist binnen de kortste keren allemaal contacten te leggen binnen het Ministerie. Ik wil jullie bedanken voor de wekelijkse meetings en ik kijk uit naar de films die jullie nog zullen maken.

Guus Berkelmans heeft ook een belangrijke rol gespeeld. Je bent wiskundig ijzersterk en ziet ontzettend snel waar het probleem zit. Meerdere malen

ben ik jouw kamer in gelopen met een wiskundig probleem of juist een idee, wat uiteindelijk leidde tot twee hoofdstukken van dit proefschrift. Ik ben trots op het onderzoek dat wij samen hebben gedaan. Verder was jij als een echte sociale vlinder bij elke borrel, bijeenkomst, en feestje. Je was er altijd. Dit vind ik een noemenswaardige goede eigenschap van jou.

Dan zijn er nog een heleboel figuranten, die nog niet zijn genoemd. Zonder hen, zou de film erg leeg zijn geweest. Daarom wil ik de *Stochastics* groep van het CWI (o.l.v. *Bert Zwart*) en de *Analytics & Optimization* groep van de Vrije Universiteit Amsterdam (o.l.v. *Ger Koole*) bedanken voor alle tafeltennis-sessies, uitjes, koffiemomenten, lunches en seminars.

Dan rest er nog één belangrijke groep, die essentieel zijn voor het maken van een film: de crew. Zonder mensen die het decor bouwen, de belichting regelen, en kostuums ontwerpen, zou het meer hebben geleken op mijn groep 8 musical. Achter de schermen hebben mijn familie, vrienden, en vriendin hiervoor gezorgd. Zonder de nodige afleiding, ontspanning, en hamburgers, had ik niet zorgeloos kunnen werken. Hiervoor zal ik ze altijd dankbaar zijn.

Het was een voorrecht om deze PhD te mogen doen.

Contents

Chapter 1

Introduction

*"Okay, Houston, we've had ~~a problem~~ **many problems** here"*

1

Data is useless without *data science*. This rapidly developing field aims to extract *knowledge* and *understanding* from any kind of data. It gives meaning to the bits and bytes in this world. In an age where we have an abundance of data, techniques from data science have had many success stories. Understanding what the data tells us is extremely useful for gaining insights and making predictions. The focus of many data scientists is on *predictive* methods for practical applications. *Classification* and *regression* techniques are able to automatically learn from data in order to make *predictions* about unseen data. In our research, we examine and improve *underexposed fundamental* parts of data science. Furthermore, we swim against the current, questioning undisputed techniques that are commonly used, and providing better alternatives. Hence, this dissertation should be read by any data scientist or analyst.

This dissertation is like a box of chocolates for any mathematician or computer scientist. It consists of six different flavors in the fields of *artificial intelligence*, *machine learning*, *statistics*, and *data analysis*. Some flavors are related, which is why we have partitioned this dissertation into three parts, each with a unique taste.

The first part of the dissertation is about *face generators* and *active learning*. A face generator is evaluated with a humanlike approach and a pioneering study is done to improve labeling of *pairwise distance* datasets that can be used to advance *face recognition* and *likeness* methods.

The second part is about benchmarking *binary classification* methods, where we introduce a new baseline approach. This baseline can even be theoretically derived for most common measures. Furthermore, we prove that it is the best baseline that does not use any feature values.

The third part consists of two important subjects in *data analysis* and *statistics*. Accurately quantifying *how dependent* one variable is *on* another variable is a fundamental part of many studies. Additionally, determining *how important* a feature is for predicting a target variable is crucial for understanding the data.

Taste is subjective, so some subjects might be more enjoyable to the reader than others, but every flavor can be tasted individually in the upcoming six chapters. To help the reader find a preferred taste, we briefly explain for each chapter what problems are faced and how we contribute to solving these problems. Nevertheless, we hope that the temptation to eat the entire box is not resisted, as the reader could discover new tasty flavors.

Chapter 2: Evaluating a Face Generator from a Human Perspective

Problem:

The website `thispersondoesnotexist.com` is truly fascinating, as a new human face is generated every time the site is refreshed. This face does not come from someone's personal photo album. Instead, a model is trained to create new faces by learning from real images. The quality of these faces is exciting and scary at the same time. That a model is able to learn how to generate such realistic images, is in my view truly one of the most impressive feats of the past years. It is becoming really hard for us humans to distinguish between a real and a fake face. Whilst reading previous literature about face generators, one thing became clear: A more humanlike macroscopic approach could give additional insights into these models. The performance of face generators is often measured using (intermediate) results of complex models. However, this does not guarantee that general human attributes (such as age and gender) are truly learned from the dataset. Do the generated images have similar human characteristics? The way that a computer 'sees' an image, is not how a human perceives the same picture. A second issue was raised due to privacy concerns. Are actual 'new' faces generated or does a generated face belong to an individual from the dataset that is used to train the model? Despite its relevance for practical applications, this has not yet been investigated in scientific literature.

Contribution:

We introduce a new two-pronged human approach to evaluate face generators, by predicting human attributes and clustering using face recognition models. In this research, we focus on the state-of-the-art StyleGAN2, although our approach can be used to evaluate *any* face generator. This makes our approach very general. We show that StyleGAN2 generates images that have the same attribute distributions as the input dataset. This means that it is able to learn general human concepts. Furthermore, we find that it generates faces that often do not belong to persons in the input dataset according to face recognition models. This is important for practical applications. Finally, we observe that adding *truncation* changes the attribute distributions towards the attributes of the latent variable, which could make it a useful 'steering rod' to generate images with specific characteristics.

Chapter 3: Active Pairwise Distance Learning for Efficient Labeling of Large Datasets by Human Experts

Problem:

In Chapter 2, we use four different *facial recognition* models to identify if generated faces belong to existing identities in the training data. Facial recognition methods are trained using *identity* datasets, which means that it is labeled for each image to which identity it belongs. However, this is not ideal for learning the *likeness* of two faces, as an identical twin and a total stranger both do not belong to the same identity, and are therefore dealt with similarly by these methods. To accurately measure how alike two faces are, we should ideally make a labeled likeness dataset. However, this is an extensive task, as labeling all pairwise combinations of faces quickly becomes infeasible, whereas an identity dataset can be scraped much more easily. Can the labeling process of a likeness dataset be made more efficient?

Contribution:

To address this issue, we look at a generalization of this problem that we name *Active Pairwise Distance Learning*, where the objective is to *actively learn* the pairwise distances between *all* instances. This is a more general problem, as *any* distance function could be used (including *likeness*). Starting with an unlabeled dataset, each round an expert determines the distance between one pair of instances. Thus, there is an important choice to make each round: 'Which combination of instances is presented to the expert?' The objective is to accurately predict all pairwise distances, while minimizing the workload of the expert. In this research, we establish upper and lower bound approximations including an update rule. The upper and lower bounds can be used to select a pair and to predict the final pairwise distance. In the experiments, we evaluate many *domain-independent* query strategies. The observations are therefore general, and the selection strategies are ideal candidates to function as baseline in future research. We show that using the criterion *max degree* consistently ranks among the best strategies. By using this criterion, the pairwise distances of a new dataset can be labeled much more efficiently. This chapter is a pioneering contribution to the field of *Active Learning*, which opens up a wealth of challenges for follow-up research.

Chapter 4: The Dutch Draw: Constructing a Universal Baseline for Binary Prediction Models

Problem:

A fundamental problem in *machine learning* is *binary classification*, where a model has to predict if an instance should be labeled as *zero* or *one*. 'Is this a picture of a cat or a dog?', 'Is this e-mail normal or spam?', and 'Is it going to rain or not?' are all examples of *binary* predictions. When developing a binary prediction model, a baseline method should be used to provide perspective on the performance of the model. Without a frame of reference, the performance of a model is essentially meaningless. An accuracy of 0.9 'feels' high, but could perhaps be easily achieved by a simple prediction strategy, depending on the dataset. Using a state-of-the-art model as baseline is a good practice, but can be hard due to parameter selection, long computational time, and the 'state-of-the-art' could change rapidly, which also make comparisons across different papers difficult.

Contribution:

In this research, we present a universal baseline method for all *binary classification* models, named the *Dutch Draw* (DD). This approach weighs a specific family of simple classifiers and determines the best classifier to use as a baseline. We theoretically derive the DD baseline for many commonly used evaluation measures and show that in most situations it reduces to (almost) always predicting either zero or one. The DD baseline is useful, as it is applicable to any binary classification problem, quickly determined without training or parameter-tuning and insightful conclusions can be drawn from the comparison. By introducing the DD baseline, we simplify and improve the evaluation process of any binary classification method.

Chapter 5: The Optimal Input-Independent Baseline for Binary Classification: The Dutch Draw

Problem:

Comparing to a baseline gives the ability to conclude that a prediction model is performing better than the baseline. However, by how much? If the baseline scores 0.4 and the model 0.8, it is not as simple as subtracting the two scores, as the performance does not have to be linear. This is an important insight. An increase in performance from 0.8 to 0.9 could be much harder to achieve than an increase from 0.2 to 0.6. While working on

the Dutch Draw, we came up with an idea to measure how much better a performance score is by weighing an *oracle* and a baseline that does not use feature values (basically the best guess). An essential part of this approach is that we establish what the 'best guess' actually is. What baseline method is best suited for this approach?

Contribution:

We examine all binary 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 *Dutch Draw baseline* is the best *input-independent* classifier (independent of feature values) for all *positional-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. This also shows that this baseline is the best option for our oracle approach, which can be used to provide additional insights into the performance of binary prediction models.

Chapter 6: The Berkelmans-Pries Dependency Function: A Generic Measure of Dependence between Random Variables

Problem:

Measuring and quantifying dependencies between random variables (RV's) gives critical insights into a dataset. This could reveal important explanatory relationships. When e.g., a specific disease is highly dependent on some variable X, it could guide researchers to find out if this can be used to discover a cure. Additionally, removing uninformative *independent* features improves the training of a model. The *Pearson correlation coefficient* is most commonly used to quantify dependence between RV's, even though it is well-recognized that this measure is essentially a measure for *linear* dependency only. Although many attempts have been made to define more generic dependency measures, there is yet no consensus on a standard, general-purpose dependency function. Several ideal properties of such a general function have been proposed, but without much argumentation.

Contribution:

We revise and discuss a list of desired properties for a dependency function.

Among other things, we identify an important misconception that the dependency function should be *symmetric*. Additionally, we introduce a new dependency function that provably meets all these requirements, whereas previous dependency functions fail to do so. Our general-purpose dependency function provides data analysts a powerful means to quantify the level of dependence between all kinds of variables. Critical insights can be acquired by using our new dependency function.

Chapter 7: The Berkelmans-Pries Feature Importance Method: A Generic Measure of Informativeness of Features

Problem:

In Chapter 6, we introduce a function to accurately measure the dependencies between random variables. However, when predicting a *target variable*, *many* variables could have an influence. Determining how *informative* each feature is improves *explainability* of the dataset. Due to complex interdependencies, it is unfortunately not as simple as measuring the pairwise dependencies between features. *Feature Importance* (FI) methods are specifically designed to measure the relevance of each feature. These techniques are becoming more important in recent years due to the need for explainability. For example, for many applications it is crucial to determine if racist or sexist biases play a role in the prediction process. FI techniques are becoming part of the toolbox of every data scientist. Over the years, a plethora of FI methods have been suggested, without a general consensus on the optimality of these methods. Even worse, a major problem with evaluating FI methods is that the ground truth FI is often unknown. This is one of the many reasons why it is hard to properly interpret the results of an FI method.

Contribution:

We introduce a new global FI approach named the *Berkelmans-Pries FI method*, which is a combination of *Shapley values* and the *Berkelmans-Pries dependency function* (as discussed in Chapter 6). We prove that our novel method has many useful properties, and that it accurately predicts the correct FI values in cases where the ground truth FI can be derived in an exact manner. Furthermore, we experimentally show for a large collection of FI methods that existing methods do not have the same useful properties. This shows that the Berkelmans-Pries FI method is a really valuable tool for analyzing datasets with complex interdependencies.

Publications contained in this dissertation:

- *Joris Pries, Sandjai Bhulai, and Rob van der Mei* (2022): "**Evaluating a Face Generator from a Human Perspective**". Published in *Machine Learning with Applications.* [137]
- *Joris Pries, Sandjai Bhulai, and Rob van der Mei* (2023): "**Active Pairwise Distance Learning for Efficient Labeling of Large Datasets by Human Experts**". Accepted for publication in *Applied Intelligence.* [136]
- *Etienne van de Bijl, Jan Klein, Joris Pries, Sandjai Bhulai, Mark Hoogendoorn, and Rob van der Mei* (2022): "**The Dutch Draw: Constructing a Universal Baseline for Binary Prediction Models**". Under revision. [14]
- *Joris Pries, Etienne van de Bijl, Jan Klein, Sandjai Bhulai, and Rob van der Mei* (): "**The Optimal Input-Independent Baseline for Binary Classification: The Dutch Draw**". Accepted for publication in *Statistica Neerlandica.* [138]
- *Guus Berkelmans, Joris Pries, Sandjai Bhulai, and Rob van der Mei* (2023): "**The BP Dependency Function: A Generic Measure of Dependence between Random Variables**". Published in the *Journal of Applied Probability.* [13]
- *Joris Pries, Guus Berkelmans, Sandjai Bhulai, and Rob van der Mei* (2023): "**The Berkelmans-Pries Feature Importance Method: A Generic Measure of Informativeness of Features**". Submitted for publication. [135]

Publications not contained in this dissertation:

- *Etienne van de Bijl, Jan Klein, Joris Pries, Rob van der Mei, and Sandjai Bhulai* (2022): "**Detecting Novel Application Layer Cybervariants using Supervised Learning**". Published in *International Journal On Advances in Security.* [15]
- *Britt van Leeuwen, Arwin Gansekoele, Joris Pries, Etienne van de Bijl, and Jan Klein* (2022): "**Explainable Kinship: A Broader View on the Importance of Facial Features in Kinship Recognition**". Published in *International Journal On Advances in Life Sciences.* [102]

Python implementations:

- The Dutch Draw: [134].
- The Berkelmans-Pries dependency function: [132].
- The Berkelmans-Pries Feature Importance method: [133].

Part I

Fabricating Faces and Labeling Likeness

Chapter 2

Evaluating a Face Generator from a Human Perspective

Contents

Based on *Joris Pries, Sandjai Bhulai, and Rob van der Mei* (2022): "**Evaluating a face generator from a human perspective**". Published in *Machine Learning with Applications*. [137]

2

Abstract

StyleGAN2 is able to generate very realistic and high-quality faces of humans using a training set (FFHQ). Instead of using one of the many commonly used metrics to evaluate the performance of a face generator (e.g., FID, IS and P&R), we use a more humanlike approach providing a different outlook on the performance of StyleGAN2. The generator within StyleGAN2 tries to learn the distribution of the input dataset. However, this does not necessarily mean that higher-level human concepts are preserved. We examine if general human attributes, such as age and gender, are transferred to the output dataset and if StyleGAN2 is able to generate actual new persons according to facial recognition methods. It is crucial for practical implementations that a face generator not only generates new humans, but that these humans are not clones of the original identities. This chapter addresses these questions. Although our approach can be used for other face generators, we only focused on StyleGAN2. First, multiple models are used to predict general human attributes. This shows that the generated images have the same attribute distributions as the input dataset. However, if truncation is applied to limit the latent variable space, the attribute distributions change towards the attributes corresponding with the latent variable used in truncation. Second, by clustering using face recognition models, we demonstrate that the generated images do not belong to an existing person from the input dataset. Thus, StyleGAN2 is able to generate new persons with similar human characteristics as the input dataset.

2.1 Introduction

Think of an unknown face. Humans are capable of imagining faces they have never seen before, combining facial attributes from multiple sources to create a new identity. Can a machine do the same? By looking at real images of humans, can it learn how to generate a unique and realistic face? And if so, are humans still able to distinguish between authentic and computer-generated faces? These questions are part of a larger quest of discovering the capabilities and boundaries of machines. Speech, music, paintings, images, and even videos are among the many things a computer is now able to generate. The quality of the produced content has increased rapidly since the introduction of generative adversarial networks (GAN [63]). Generating realistic faces shows the power, capabilities, and limitations of these approaches.

In 2019, the successor [84] of the well-known StyleGAN [83] paper was published. When StyleGAN [83] was released in 2018, it immediately showed impressive results. At that time, this architecture improved the state-of-the-art performance considerably by injecting the generator at different stages with a style-based latent variable. Although humans can still distinguish between computer-generated and real images, the images look very realistic at first glance. This is a huge achievement. Especially if one considers that only since 2017, Karras et al. [82] were able to generate high-resolution images (1024×1024 pixels). Only small details give away that these images are not real [185]. The successive paper [84] claims to improve the images even further, making them even less distinguishable. Their new approach is called StyleGAN2.

As the name suggests, StyleGAN2 is trained using a generative adversarial network (GAN) [63]. The basis of this approach is to let two models compete against each other, making each model better at their specific task. More specifically, one model tries to generate images that resemble real faces, whereas the other model tries to distinguish between the real and the generated images. The generator within StyleGAN2 tries to learn the distribution of the input images, which is monitored using the metrics FID, P&R, and PPL. According to these metrics, StyleGAN2 is successful in learning the input distribution. However, this does not necessarily mean that higher-level human concepts are preserved. Are the input and generated images similar from a human perspective?

When a human compares two faces, common measures for evaluating GANs

like FID [70], IS [150] or PPL [83] are not natural, as these measures are artificially using e.g., a neural network to evaluate the performance. This is not a human approach, a person would rather compare human characteristics to evaluate the images. Although it is infeasible to compare a lot of generated images by hand, a humanlike approach is necessary. Zhou et al. [199] did use humans to decide whether images generated by StyleGAN were fake or real. The results showed that StyleGAN was capable of generating faces that were hard to distinguish by humans from the input images. Our research focuses on two different aspects: Are human traits transferred from the input to the output dataset and are the generated images new identities? Lack of attribute and identity labels tagged by humans for StyleGAN2, leads us to use existing models that were trained using different humanly labeled data.

Hence, we take two separate paths to evaluate how well StyleGAN2 performs from a 'human' perspective. First, multiple models are used to predict general attributes of the images, such as age, gender, and race. In this way, we can determine if higher-level concepts are preserved. Second, we examine for multiple *face recognition* models if the generated images can be considered to be different persons, compared to the images from the input dataset.

With this two-pronged approach, we are able to show that the human attribute distributions are very similar for the input and output dataset, but the generated images are nonetheless different according to the facial recognition models. Thus, StyleGAN2 has the best of two worlds. It is able to copy high-level concepts from the input dataset, whilst still creating different persons. Furthermore, if *truncation* (see Section 2.2.1) is used to limit the latent variable space, the attribute distributions change significantly towards the attributes corresponding with the latent variable used in truncation. To our knowledge, this humanly approach to comparing high-level concepts of facial datasets is new. While we will only use our two-pronged approach for StyleGAN2, it can also be used to evaluate other face generators.

To summarize, in this chapter we:

- introduce a new two-pronged humanly approach to evaluate face generators, by predicting human attributes and clustering using face recognition models;
- show that the state-of-the-art StyleGAN2 generates images that have the same attribute distributions as the input dataset;

- determine that StyleGAN2 generates faces that often do not belong to persons in the input dataset according to face recognition models;
- observe that adding truncation to the latent variable space changes the attribute distributions towards the attributes corresponding with the latent variable used in truncation.

The remainder of this paper is organized as follows. First, the relevant datasets are discussed in Section 2.2. Next, in Section 2.3 the methods are explained that are used to predict facial attributes, embed faces, and cluster on these embeddings. Furthermore, we also define how a cluster is evaluated and why clustering is a natural approach. The results are discussed in Section 2.4. Finally, Section 2.5 summarizes the general findings and discusses possible future research opportunities.

2.2 Datasets

Karras et al. [84] made three datasets publicly available that are used in this research: The input dataset (FFHQ), consisting of 70,000 real facial images (without identity annotation); Two output datasets of StyleGAN2, both consisting of 100,000 generated images. The only difference in the creation of these output datasets is the so-called *truncation* [23, 83, 84] parameter. All images are high-quality pictures (1024×1024 pixels).

2.2.1 Truncation

To explain how truncation works, it is useful to take a look at the structure of StyleGAN (see Figure 2.1). Note that there are some differences with the architecture of StyleGAN2. However, the following core principles still hold. Some latent variable $\boldsymbol{z} \in \mathcal{Z}$ from latent space $\mathcal{Z}$ goes into a mapping network f, after it is normalized using pixelwise feature vector normalization [81]. This results in a different variable $\boldsymbol{w} \in \mathcal{W}$ such that $f(\boldsymbol{z}) = \boldsymbol{w}$. $\mathcal{W}$ is the so-called *intermediate latent space*. Next, the expectation of the intermediate latent variable is determined by $\bar{\boldsymbol{w}} := \mathbb{E}_{\boldsymbol{z} \sim P(\boldsymbol{z})}[f(\boldsymbol{z})]$, where $P(\boldsymbol{z})$ is the probability that $\boldsymbol{z}$ is randomly drawn from $\mathcal{Z}$. The authors of [83] state that $\bar{\boldsymbol{w}}$ represents "a sort of an average face".

$\bar{\boldsymbol{w}}$ is used to truncate the intermediate latent space. Given a $\boldsymbol{w} \in \mathcal{W}$, truncation returns a different intermediate latent variable, denoted $\boldsymbol{w'}$, such that $\boldsymbol{w'} = \bar{\boldsymbol{w}} + \psi \cdot (\boldsymbol{w} - \bar{\boldsymbol{w}})$, where $\psi \in \mathbb{R}$ is called the *truncation parameter*. Note that $\psi = 1$ gives $\boldsymbol{w'} = \boldsymbol{w}$, which is the same as not applying truncation at all. In Figure 2.2, five faces are shown that are generated with $\bar{\boldsymbol{w}}$ as

intermediate latent variable. This is equivalent to generating images with $\psi = 0$. Furthermore, noise is injected in the synthesis network to increase stochasticity, see Figure 2.1. However, this leads to only minor changes if the intermediate latent variable is constant. As can be seen in Figure 2.2, the faces all look very similar.

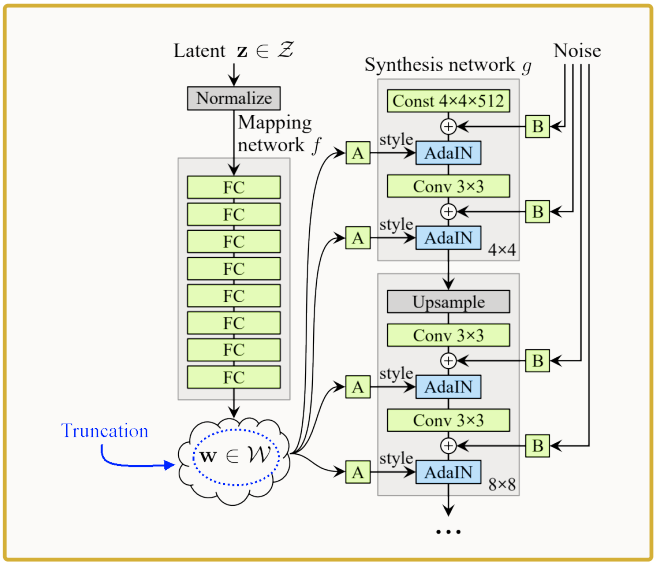


Figure 2.1: StyleGAN structure: Architecture of the StyleGAN approach (extracted from [83]). Truncation limits the intermediate latent space $\mathcal{W}$.

Figure 2.2: Average intermediate latent space: These faces (seeds 0000-0004) are generated using the expected intermediate latent variable $\bar{\boldsymbol{w}}$.

2.3 Methodology

To compare the input images of a face generator with its output, two separate paths are taken. First, multiple models are used to predict human attributes. This allows for a high-level comparison between the different datasets. Are

characteristics, such as age and gender, the same for the input and output datasets? Secondly, clustering using face recognition models could determine if the generated faces belong to an existing person from the input dataset. Do the output datasets consist of different persons, or are they embedded similarly compared to the input dataset? Combining these two approaches gives a clear view of the performance of a face generator.

The output of StyleGAN has already been examined to some extent. Karras et al. [84] evaluated the generated images in order to eliminate artifacts. For different datasets, FID and PPL was compared between StyleGAN and StyleGAN2 [83]. Furthermore, efforts have been made to understand and steer the latent space [159]. By manipulating the latent space, one could change certain attributes of an image. For example, Shen et al. [159] showed that it is possible to alter the age, gender, smile, pose, and add or remove eyeglasses. However, to our knowledge, our humanly approach to comparing high-level concepts of facial datasets is new. While we will only look at datasets from StyleGAN2, our two-pronged approach can also be used to evaluate other face generators. Within our novel approach, existing methods are used for predicting, embedding and clustering. These methods will all be discussed in the upcoming sections. A general overview of our proposed approach can be found in Figure 2.3.

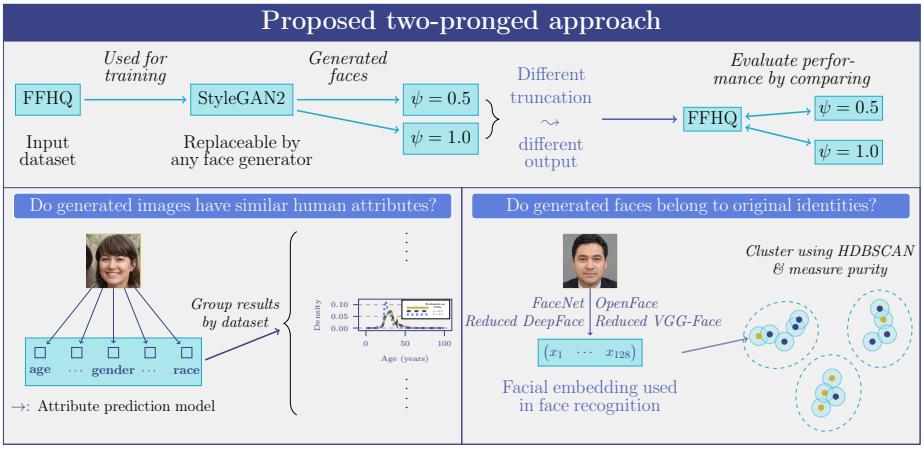


Figure 2.3: General concept: A general overview of our proposed two-pronged approach. Using different truncation parameters, a comparison is made (with attribute prediction models and facial embedding methods) between the input and output datasets in order to determine if generated images have similar human attributes and if they belong to original identities.

2.3.1 Attribute prediction

A face has many characteristics. This leads to a wide variety of attribute predictions: pose [195], skin color [178], and even attractiveness [190], to name just a few. We select the following group of features to examine: *age*, *gender*, *race*, *horizontal rotation*, and *vertical rotation*. Note that these features cover general human concepts, but additional attribution models can always be added to or removed from this framework. To clarify what we mean by 'human concept', we argue that every identifiable aspect of a face (or body) that has been given a name can be called a human concept. For example, the eyebrow is identified by humans as a specific part of the face. But also, somebody can look young or old. We consider these examples as 'human concepts', because we abstract information from a group of pixels with respect to some convention. In our view, any model that predicts a general human attribute, trained with humanly labeled data, could give some insight into the difference between the input and output dataset. Adding more attribute models does give additional information, but to show how our approach works, we limit ourselves to the attribute prediction models that we introduce in the following sections. Note that adding or removing other attribute prediction models does not affect the results of an individual attribute model, as each model is assessed separately.

Predicting age, gender, and race

One of the main guiding papers for this research is the *Diversity in Faces* paper by Merler et al. [116]. The aim of their research was to create an annotated dataset in order to improve the accuracy of face recognition and increase the facial diversity within commonly used datasets. Lack of diversity could harm the effectiveness of face recognition in practical implementations. It could even be discriminatory against minorities [25]. Merler et al. [116] use different models to annotate images from the YFCC-100M dataset [175]. These models predict a plethora of attributes for each face. The same kind of models, implemented in *deepface* [153], are used to predict the *age*, *gender*, and *race* of a person.

However, there is a difference between the implementation of *deepface* [153] and the prediction models from [116]. *deepface* uses the VGG-Face neural network [126], whereas [116] follows the approach of Rothe et al. [145], who use the VGG-16 architecture [162]. The VGG-Face network [126] is specifically trained to recognize faces, whereas the VGG-16 network is trained with ImageNet [146] by Rothe et al. [145]. ImageNet contains a wide variety of images, not limited to faces. This is why we decided to follow *deepface*

and use the VGG-Face network.

For each attribute, a similar procedure is followed. *deepface* uses a pre-trained VGG-Face network [126] as the starting point. Only the last few layers are replaced and retrained to fit the objective. There are some important details about these models (see [153] for technicalities):

- Counterintuitively, age prediction is not made using regression. Rothe et al. [145] claim that using classification instead of regression improved the performance and also stabilized the training process. The output layer consists of 101 variables, each corresponding to an age in years (0-100). The last layer has a softmax activation function, which ensures that the output of the last layer is a probability distribution over the different output variables. The age is finally predicted by taking the expectation over these output variables, see Rothe et al. [145].
- Gender prediction is made using two output variables, corresponding to *woman* and *man*.
- For race prediction, a distinction is made between the following races: *Asian*, *Indian*, *Black*, *White*, *Middle Eastern*, and *Latino Hispanic*.
- *deepface* uses the *haarcascade frontalface default* detector from OpenCV [21] to center, trim, and resize an image. However, it can occur that the facial detector does not recognize a face. When this happens, the image is simply omitted from the analysis of the corresponding attributes.

Serengil and Ozpinar [153] self-reported on the performance of the models. The mean absolute error of the age model was 4.65 and the accuracy of the gender model was 97.44% with 96.29% precision and 95.05% recall. However, the models were not evaluated on the datasets that will be used in this research, because there exist no annotated labels of these features yet. It is therefore unclear how well these models perform for the datasets that are used. Nevertheless, we want to stress the fact that these models are only used to compare the characteristics of each dataset globally. Even if the models perform worse (due to domain shift), they can still be insightful for comparing the datasets.

Predicting horizontal and vertical rotation

To measure the horizontal and vertical rotation, *dlib* [88] is used. It can predict the position of 68 general landmarks on a face (see Figure 2.4). These landmarks can be used to crop an image or measure attributes such as face

and nose width/height. We use the landmarks to estimate the horizontal and vertical position of a head. It must be noted that these points remain a prediction. Especially when a head is rotated too much, these predictions lose accuracy.

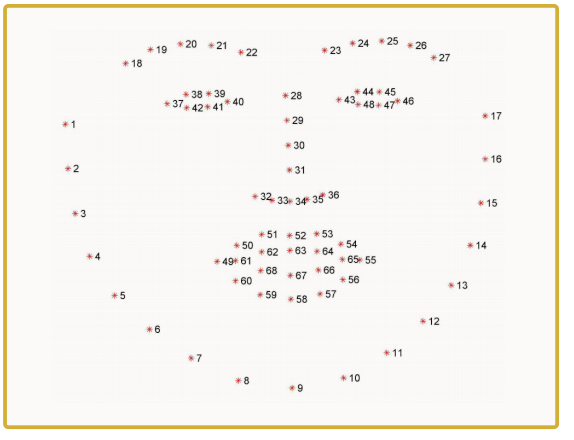


Figure 2.4: Dlib landmarks: *Dlib* predicts the coordinates of these 68 landmarks for each face (extracted from [149]).

There are many ways to estimate the horizontal rotation (yaw) and vertical rotation (pitch) of a head [22, 45, 195]. However, we are mainly interested in the differences between the datasets. Therefore, we are not much concerned about obtaining the best accuracy for each individual image. Thus, we use a simple concept to estimate the horizontal and vertical rotation, see Figures 2.5 and 2.6.

Let x_i, y_i denote the horizontal and vertical position of landmark i, respectively. Observe that when a head rotates sideways, the horizontal distance between the tip of the nose and the corner of the eyes changes. To scale this measure properly, this distance is compared with the horizontal distance between both lateral eye corners. Thus, the fraction

$$\frac{|x_{\text{right lateral eye corner}} - x_{\text{nose tip}}|}{|x_{\text{right lateral eye corner}} - x_{\text{left lateral eye corner}}|} \tag{2.1}$$

is measured to approximate horizontal rotation (see Figure 2.5). When a head is straight, the tip of the nose is assumed to be in the middle. But, when it rotates to a side, the fraction becomes smaller or larger, depending on the side it is rotating towards. Note that this fraction could be used to approximate the horizontal rotation in degrees using known facial rotations.

However, varying nose shapes and facial asymmetries could influence the results.

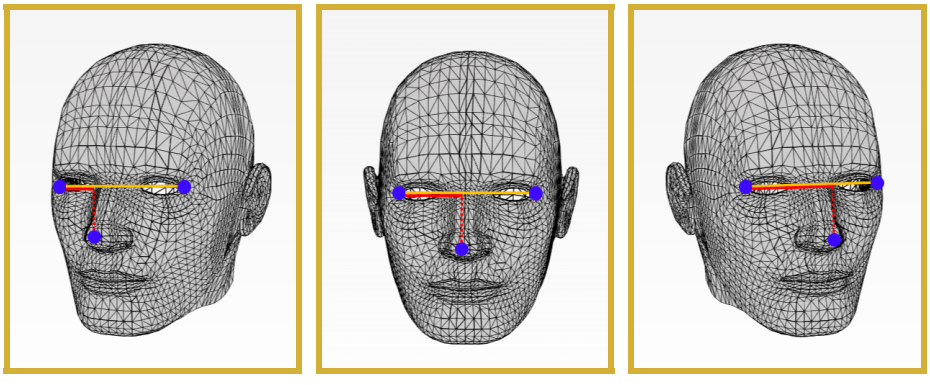


Figure 2.5: Horizontal rotation: Horizontal rotation is measured by dividing the red bar by the yellow bar (see Equation (2.1)). The positions of the blue dots are predicted by *dlib*.

Using a similar key insight, vertical rotation can be measured by comparing the vertical distance between the nose root and nose tip and the vertical distance between the nose root and the chin. Thus, the fraction

$$\frac{|y_{\text{nose root}} - y_{\text{nose tip}}|}{|y_{\text{nose root}} - y_{\text{chin}}|} \tag{2.2}$$

is measured to approximate the vertical rotation, see Figure 2.6. Note that this measure is more subjective to personal traits, as nose lengths can vary. Although this may raise issues for an individual image, we believe that this method is sufficient for comparing the datasets generally, as individual errors will not have a large impact on the general comparison.

2.3.2 Facial embedding with face recognition models

Are new individuals created or are the generated images too similar to individuals from the input dataset? To compare the images from a human perspective, some kind of *facial embedding* is necessary. It is imperative that the dimensionality of each image is reduced. Every image consists of 1024×1024 pixels and each pixel consists of three color values (RGB). Given the size of these datasets, it is unfeasible to compare the pixels for each pair of images. Furthermore, a human does not compare two images pixel by pixel. Instead, one matches facial features such as eyes, nose, hair, and mouth to evaluate if these two images belong to the same person. This is

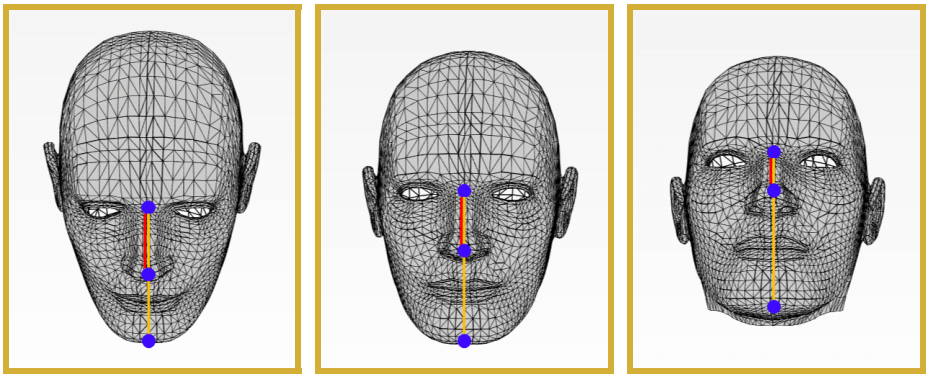


Figure 2.6: Vertical rotation: Vertical rotation is measured by dividing the red bar by the yellow bar (see Equation (2.2)). The positions of the blue dots are predicted by *dlib*.

why we decide to use face recognition models, where a face is first embedded to a point in a latent space, such that distances can be measured between faces. If two points are close, they are assumed to be similar. In this way, we can determine if new individuals are created. The four facial embedding methods used for facial recognition are outlined below.

FaceNet [152] is well suited for our objective. It is a deep convolutional network that converts an image (160x160 pixels) to a 128-dimensional vector that lies on the 128-dimensional hypersphere. To find an appropriate embedding, *FaceNet* uses a triplet loss function.

OpenFace [7] follows the same concept as FaceNet [152]. It is, however, open-source and focuses on real-time face recognition. It converts an image (96×96 pixels) to a 128-dimensional vector that lies on the 128-dimensional hypersphere.

DeepFace [173] uses 3D face modeling and a large deep neural network to recognize faces. It converts an image (152×152 pixels) to a 4096-dimensional vector, which is then used to identify individuals using a classification layer. Taigman et al. [173] call this vector the "raw face representation feature vector". **VGG-Face** [126] uses the well-known VGG-16 architecture [162] to specifically train for facial recognition. It converts an image (224×224 pixels) to a 2622-dimensional vector. This model also uses the specific loss function from FaceNet [152] to train the model for facial recognition.

Dimensionality reduction

The output vector of DeepFace [173] and VGG-Face [126] is too large to properly cluster on. Therefore, the dimensionality is reduced with *singular value decomposition* (SVD) from a 4096- and 2622-dimensional vector to a 128-dimensional vector. Thus enforcing the same dimensions of the output vector for each embedding method. This dimensionality reduction could weaken the accuracy of these models, as some information is lost. However, if there is still a clear distinction between the datasets in this lower dimension, there must be a similar or larger, difference in the higher dimension. We call these models *Reduced DeepFace* and *Reduced VGG-Face* from now on.

2.3.3 Clustering

Once the faces are embedded using face recognition models, images can be compared. There are many options, however we will show why clustering is the most natural approach in our view. In the end, we want to answer the question if actual new persons are generated. The face recognition methods enable us to measure the distance between each pair of images. If the distance between images A and B is below a defined threshold, the images are considered to be of the same identical person. However, if the distance between images B and C is also below the threshold, images A, B and C all belong to the same person and form a cluster. Thus, a clustering approach naturally arises by this logic. Each cluster of images represents a single person, according to the face recognition methods.

To investigate if the output dataset contains the same identities as the input dataset, two combinations are made:

- **FFHQ $\cup$ ($\psi = 1$):** The input dataset combined with the generated images without truncation;
- **FFHQ $\cup$ ($\psi = 0.5$):** The input dataset combined with the generated images with truncation.

The clustering is done on the embeddings of these two combinations. Due to the size of the datasets (170,000 images in total), a clustering method with few parameters is preferred. Furthermore, there is no or little domain knowledge of proper parameter values, making most clustering methods too computationally expensive, as a range of values for the parameters needs to be evaluated.

This leads to the decision to use *HDBSCAN* [27]. The idea behind this algorithm is that instances A and B are *neighbors* if the distance between

them is less than or equal to ϵ and two instances A and B are in the same cluster if there exists a sequence of instances from A to B such that each successive instance is a neighbor of the previous. *HDBSCAN* allows ϵ to be altered post-completion. In this research, we use the implementation of [115] with the *Euclidean* distance function. Although *HDBSCAN* has computational complexity $\mathcal{O}(n^2)$ [27] with n the number of samples, McInnes et al. [114] show that *HDBSCAN* performs reasonably fast for large datasets. Furthermore, it returns a hierarchical clustering. This is useful to determine different statistics post-completion. If instead the very similar *DBSCAN* [50] is used, some information about the parameter ϵ is necessary. ϵ determines the neighborhood of each point. The relevant range for ϵ varies greatly for different embeddings. Without large computational costs, it is possible to determine the results for different values of ϵ using the hierarchical cluster, after running *HDBSCAN*.

HDBSCAN has a single primary input parameter m_{pts} [27]. This parameter determines if a group of samples is large enough to be considered an actual cluster. If two images are embedded closely together, they should be able to form a cluster, as they can belong to the same person. Thus, $m_{pts} = 2$ is a natural choice, as it allows all cluster sizes except a cluster containing a single image.

Cluster evaluation

The goal of clustering is to investigate if the input and output datasets consist of different individuals. Therefore, *purity* [112] is used to measure the intertwinedness of the clustering, as this metric evaluates if subclusters consist of purely real or generated images. *Purity* is measured by counting the samples of the most frequent class in each cluster and dividing by the total number of samples. More formally, let clustering C of N samples consist of subclusters C_i for $i \in \{1, \ldots, K\}$, for some $K \in \mathbb{N}_{>0}$. Each sample j comes from a corresponding dataset labeled l_j. For each subcluster C_i, let d_i denote the label of the dataset that occurs most frequently, then:

$$\text{Purity}(C) = \frac{1}{N} \cdot \sum_{i=1}^{K} \sum_{j \in C_i} \mathbb{1}_{l_j = d_i}. \tag{2.3}$$

If $\text{Purity}(C) = 1$, it means that every subcluster only contains samples of one class. If there are only two classes, a lower bound of *purity* is $\text{Purity}(C) = 0.5$, as in the worst case every subcluster is split 50/50 between the classes.

Baseline purity Note that the upper and lower bound, previously given, cannot always be achieved. This is dependent on the distribution of the labels and the structure of a clustering. For example, if there is only one cluster and the labels are divided 80/20, the purity score will be 0.8. Therefore, a better baseline is necessary to evaluate how good/bad a purity score of a clustering is.

Assume that the input and output dataset are sampled from the same distribution. Then there is no way of telling which image is drawn from which dataset. For each parameter combination, *HDBSCAN* returns a cluster with a certain structure. Each cluster consists of a number of subclusters all with a corresponding size. If there would be no difference between the two datasets, it would correspond with randomly assigning each sample to a position in the cluster. As we have seen before, the structure of the cluster is important for the purity score. Therefore, we approximate the expected purity score of a randomly assigned cluster with the same structure as provided by *HDBSCAN*. Under the hypothesis that there is no difference between the datasets, we get an average purity score that is ultimately used to compare the results. If the results are close to this baseline, it means that the datasets are very similar. On the other hand, if there is a clear distinction between the baseline and the results, it would mean that the datasets are not alike.

2.4 Analysis

The images from the datasets are analyzed in two ways. First, models are used to predict certain attributes of each face (e.g., gender and age). This will determine the distribution of these features, which can be used to compare the datasets globally. Second, multiple embedding methods are used in combination with a clustering method. By looking inside each subcluster and evaluating the purity score (see Section 2.3.3), a comparison between the datasets can be made. The results of both approaches are discussed below.

2.4.1 Results attributes

For each image in every dataset, models were used to predict the following attributes: *age*, *gender*, *race*, *horizontal rotation*, and *vertical rotation*. The results are grouped together per dataset. This gives a global overview of these attributes for each dataset. In particular, we are interested in the similarity of the distributions. If these distributions are different, it would

suggest that the underlying datasets are in fact different.

Results age

Without truncation ($\psi = 1$), the age distribution of the generated images is almost identical to the input dataset (FFHQ), see Figure 2.7. Even the small peak around 40 years is similar for these two datasets. If truncation is added ($\psi = 0.5$), it can be observed that the distribution shifts more towards the younger age groups.

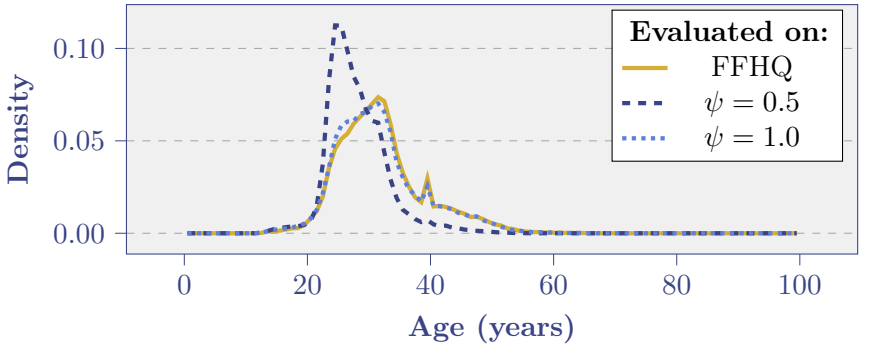


Figure 2.7: Age distribution: Age distribution averaged per dataset.

Results gender

The model returns for both classes (*woman* and *man*) a probability value. The *dominant gender* is the gender with the highest probability of the two. Note that without truncation ($\psi = 1$), the distribution is almost the same as the input dataset (FFHQ), see Figure 2.8. Whereas with truncation ($\psi = 0.5$), relatively more females are generated compared to the input.

Results race

Table 2.1 shows the average probability mass for each race. Table 2.2 shows the distribution of the *dominant race*. This is the class that obtained the maximum probability given by the model. Without truncation ($\psi = 1$), the distribution is very similar to the input dataset (FFHQ). However, if truncation is added ($\psi = 0.5$), *white* is predicted more often.

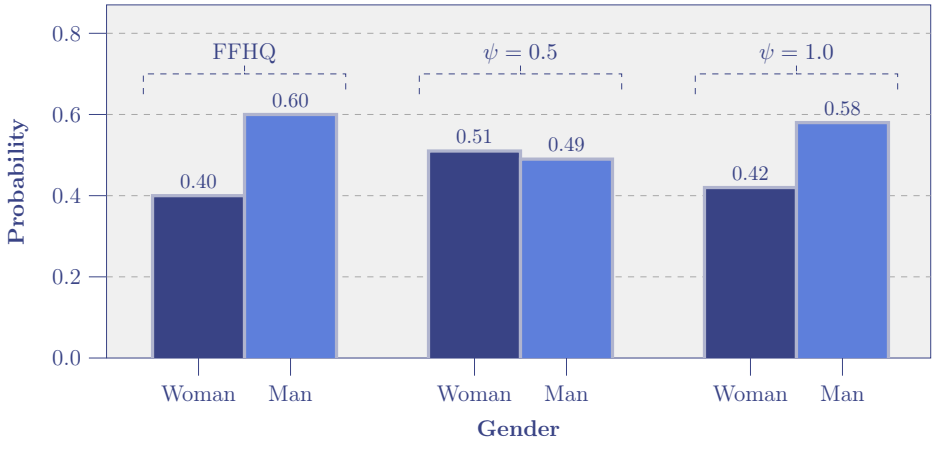


Figure 2.8: Dominant gender: Probability averaged per dataset that *man* or *woman* gets the highest prediction probability.

Table 2.1: Average probability: Race distribution averaged per dataset.

Dataset	Race					
	Asian	Indian	Black	White	Middle Eastern	Latino Hispanic
FFHQ	0.1826	0.0425	0.0549	0.4889	0.0965	0.1346
$\psi = 0.5$	0.0921	0.0198	0.0118	0.6879	0.0805	0.1079
$\psi = 1$	0.1883	0.0366	0.0579	0.4900	0.0933	0.1339

Table 2.2: Dominant race probability: Probability averaged per dataset that a race class gets the highest prediction probability.

Dataset	Race					
	Asian	Indian	Black	White	Middle Eastern	Latino Hispanic
FFHQ	0.1961	0.0183	0.0509	0.5775	0.0513	0.1058
$\psi = 0.5$	0.1031	0.0034	0.0084	0.7769	0.0346	0.0735
$\psi = 1$	0.2066	0.0100	0.0552	0.5719	0.0501	0.1062

Results horizontal rotation

As explained by Figure 2.5, the horizontal rotation is measured using the predicted landmarks of *dlib* (see Equation (2.1)). In Figure 2.9, it can be observed that without truncation ($\psi = 1$), the distribution is nearly identical. When truncation is added, the distribution narrows to 0.5, which means that more straight faces are generated or the faces are more symmetric.

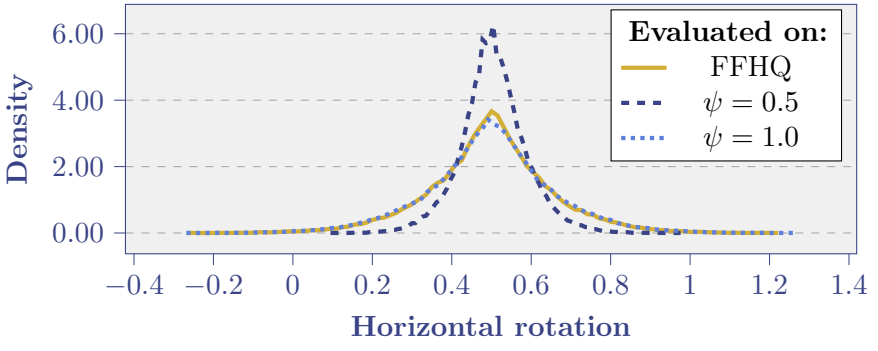


Figure 2.9: Horizontal rotation: Horizontal rotation averaged per dataset, predicted using the landmarks of *dlib* (see Figure 2.5).

Results vertical rotation

As explained by Figure 2.6, the vertical rotation is measured using the predicted landmarks of *dlib* (see Equation (2.2)). Again, the distribution without truncation ($\psi = 1$) is identical to the distribution of the input dataset FFHQ (see Figure 2.10). If truncation is added ($\psi = 0.5$), the distribution shifts to the right. There are two possible explanations. First, it could mean that the generated images are rotated more downwards. Second, it is possible that the generated images have a longer nose. In Figure 2.11, the distributions of the nose length can be found. There is a significant shift when truncation is added ($\psi = 0.5$). Thus, it can be concluded that the nose lengths are on average larger for $\psi = 0.5$.

Failed detections deepface

The models that predict the age, gender, and race were trained using a specific face detector. When the detector finds a face, it automatically trims and resizes the image. However, this detector sometimes fails to detect a face. In this case, the image is simply omitted from the attribute analysis. Figure 2.12 shows how often the detector is successful. Note that with truncation ($\psi = 0.5$) this failure probability decreases drastically. The

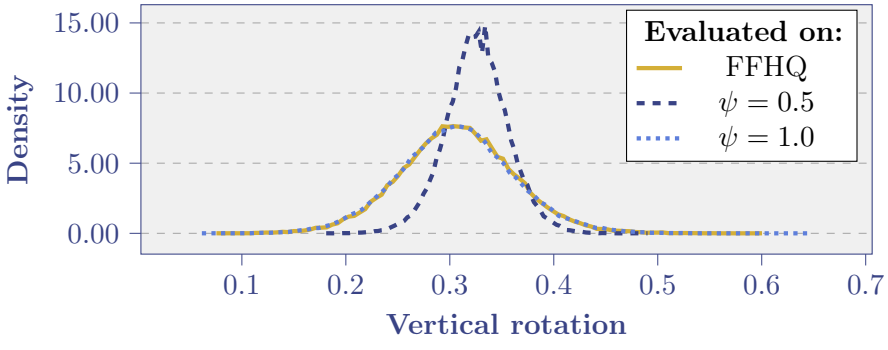


Figure 2.10: Vertical rotation: Vertical rotation averaged per dataset, predicted using the landmarks of *dlib* (see Figure 2.6).

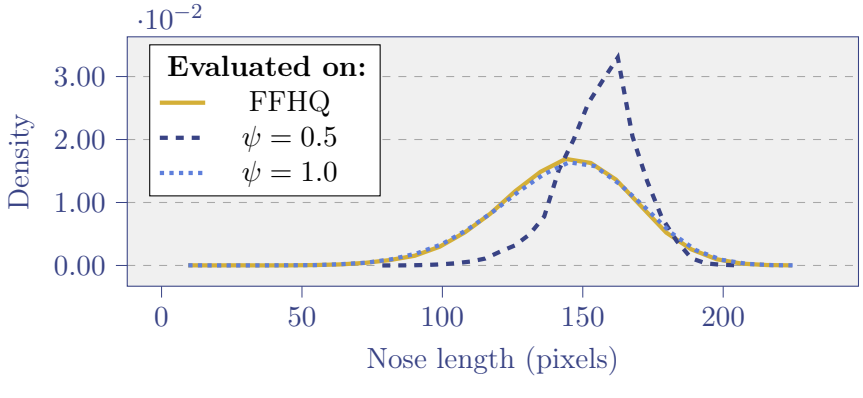


Figure 2.11: Nose length: Nose length averaged per dataset, estimated by measuring the distance between the nose root and nose tip.

results for FFHQ and no truncation ($\psi = 1$) are similar and relatively high. A failure rate of around 10 percent is rather substantial.

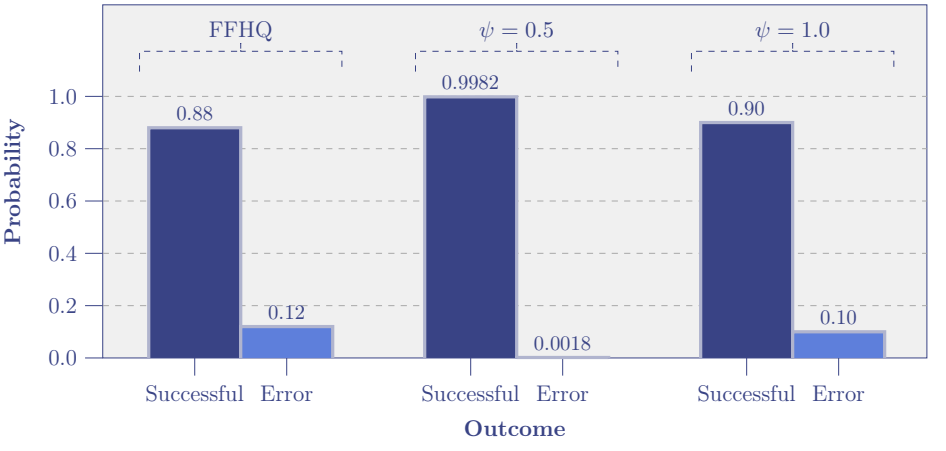


Figure 2.12: Failure rate deepface detector: Probability averaged per dataset that the *haarcascade detector* [21] (used in *deepface*) fails to detect a face.

In Figures 2.13a to 2.13c, the first images of each dataset are shown where the *deepface* detector fails. Only for $\psi = 0.5$, it is not really clear why these images fail. However, we suspect that the following factors contribute to the general failure of the detector:

- eyewear;
- headwear;
- rotated heads;
- multiple persons;
- young age;
- obstructed eyes;
- structural errors (deformation, glitches, missing parts, etc.).

Note that these are only visual observations and should be investigated further.

Failed detections dlib

Dlib uses another face detector. The failure rate of this detector is also measured. As can be seen in Figure 2.15, the failure probability is much

(a) **FFHQ** (b) $\psi = 0.5$ (c) $\psi = 1.0$

Figure 2.13: deepface detector failures: The first images per dataset, where the *deepface* detector does not detect a face.

lower compared to Figure 2.12. It is notable that if truncation is added ($\psi = 0.5$), the failure probability is even zero. However, the differences between the probabilities are so small that it is hard to draw any meaningful conclusions for the different datasets. The failure rate is very small for each dataset.

In Figures 2.14a and 2.14b, the first images of each dataset are shown, where the *dlib* detector fails. Note that for $\psi = 0.5$, there are no failures. The same elements we observed in the failures of the *deepface* detector are prevalent in the *dlib* detector failures. However, the *dlib* detector seems to be more robust compared to the *deepface* detector.

2.4.2 Results clustering

In Section 2.3.3, it is discussed why clustering is a natural approach to determine if the newly generated images belong to an existing person. The clustering results can be seen in Figure 2.16. Note that different parameter values of ϵ are relevant for each embedding method. This makes *HDBSCAN* [27] very useful, as the parameter value of ϵ can be changed post-computation. Given the parameters, *HDBSCAN* returns a cluster. Two measures are of our interest. First, the number of subclusters within each cluster. This indicates how many unique persons exist in the data according to the embedding methods. Second, the *purity* of a cluster is measured (see Section 2.3.3). We cluster on the combination of the input dataset (FFHQ) and the output

(a) **FFHQ** (b) $\psi = 1$

Figure 2.14: Dlib detector failures: The first images per dataset, where the *dlib* detector does not detect a face. $\psi = 0.5$ resulted in no detection failures.

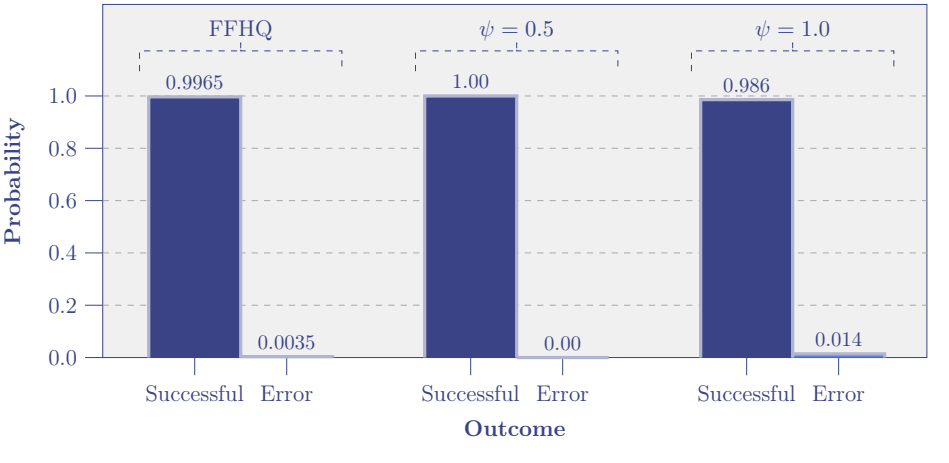


Figure 2.15: Failure rate dlib detector: Probability averaged per dataset that the *frontal face detector* (used in *dlib*) fails to detect a face.

dataset with either no truncation ($\psi = 1$) or with truncation ($\psi = 0.5$). In this way, the output dataset can be compared with the input dataset.

For each facial embedding the maximum number of subclusters (Table 2.3) is determined with $m_{pts} = 2$ (see Section 2.3.3).

Table 2.3: Maximum subclusters: For each dataset combination and facial embedding method, the maximum number of subclusters is determined with $m_{pts} = 2$.

Facial embedding	Dataset combination	
	FFHQ $\cup$ ($\psi = 1$)	FFHQ $\cup$ ($\psi = 0.5$)
FaceNet	5170	5592
OpenFace	2835	3598
Reduced DeepFace	2885	3153
Reduced VGG-Face	2865	2246

Purity results

The purity results for $m_{pts} = 2$ are shown in Figure 2.16. The relevant range for ϵ is chosen based on the number of clusters. Two main conclusions can be drawn from these graphs. First of all, 7 out of 8 clusterings show a clear distinction between the baseline and the actual purity score. Only OpenFace without truncation ($\psi = 1$) shows no obvious separation. Therefore, it can be concluded that there is a definite difference between the input and the output datasets. Thus, the generated images belong to different persons compared to the input dataset, according to the facial recognition methods. Second, the gap between the baseline and the actual purity score is much larger with truncation ($\psi = 0.5$) than without truncation ($\psi = 1.0$). Thus, truncation makes it more likely that a cluster is predominantly real or generated.

2.5 Discussion and conclusion

We presented a general two-pronged approach that tries to humanly compare the input and output datasets for a given face generator. However, we explicitly applied this approach to the state-of-the-art generator StyleGAN2 [84]. We started by comparing the input dataset (FFHQ) and the output datasets based on their attributes. Multiple models were used to predict

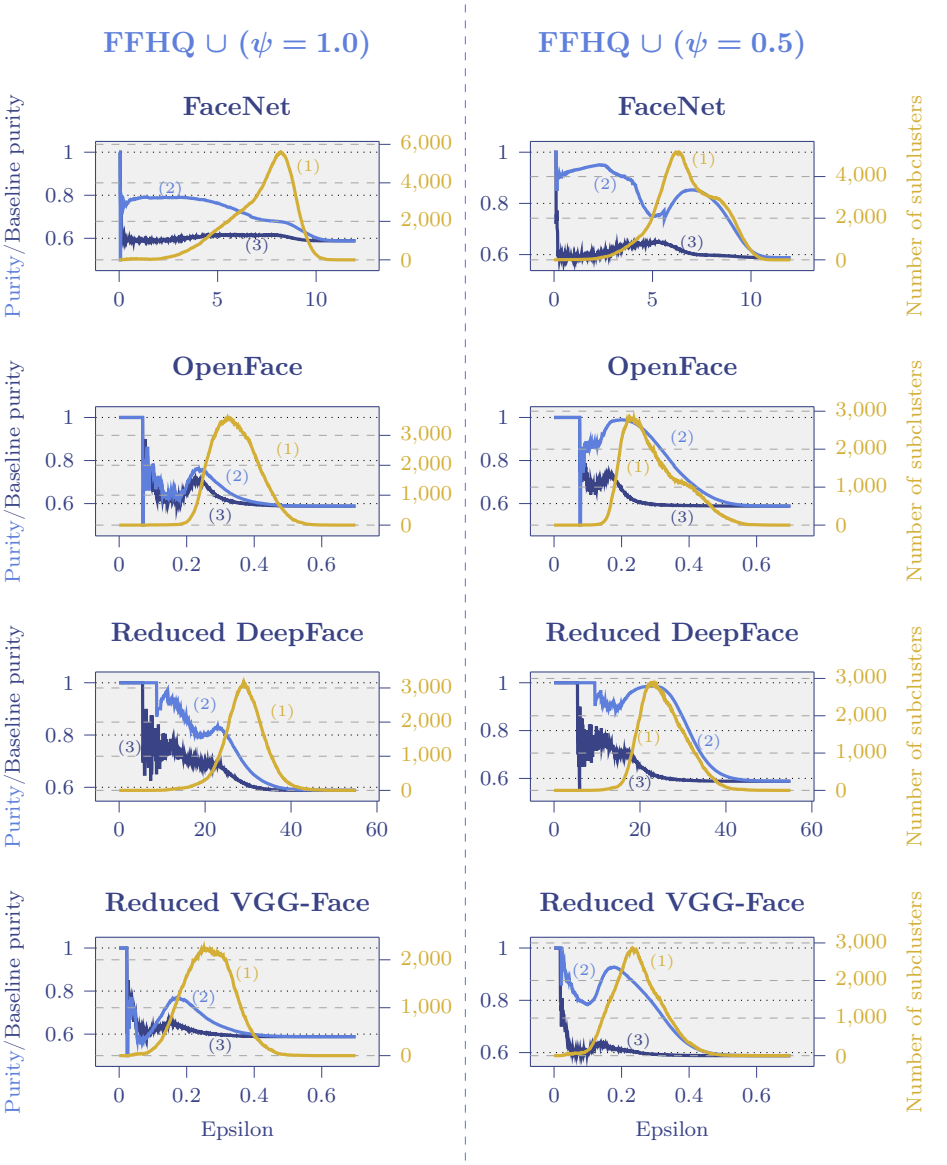


Figure 2.16: Clustering purity: Using *HDBSCAN* with $m_{pts} = 2$, ϵ determines which clustering is made. The gold line (1) denotes the number of subclusters of each cluster. The light blue line (2) is the purity score (see Section 2.3.3). The dark blue line (3) shows the approximated purity score under the hypothesis that the two datasets are similarly distributed using 100 simulations (see Section 2.3.3). The left side is the combination FFHQ $\cup$ ($\psi = 1$), whereas the right side is the combination FFHQ $\cup$ ($\psi = 0.5$).

attributes (*age*, *gender*, *race*, *horizontal*, and *vertical rotation*) for each image. The results were very clear. The attribute distributions were the same for the input dataset and the generated images without truncation ($\psi = 1$). However, when truncation is added ($\psi = 0.5$), the attribute distributions shift significantly towards the attributes corresponding with the latent variable used in truncation. Although there exist many evaluation measures for GANs [18], the three most commonly used measures are: *Fréchet Inception Distance* (FID), *Inception Score* (IS), and *Precision and Recall* (P&R) [19, 161]. FID measures the difference between the input and output images by embedding them into the feature space of an Inception Net (trained on ImageNet) [18]. IS also uses the Inception Net to measure the diversity of the generated images compared to the mean. P&R quantifies how similar the generated images are to the input dataset and how well the entire training dataset is covered. Additionally, StyleGAN2 [84] evaluates the *perceptual path length*, (PPL) which measures the difference between the VGG-16 embedding [162] of two consecutive images, where a path in the latent space is subdivided into linear segments [83]. These measures have been used to evaluate the performance of StyleGAN2 [84]. Thus, the observation that StyleGAN2 is able to learn the input dataset is not new. It is known that GANs are able to learn the input distribution, although training sometimes appears successful, whilst the target distribution is actually far from the trained distribution [9]. However, it has not previously been shown that higher-level human concepts are also preserved. It could be that somehow these measures indirectly assess these human concepts, although this has not yet been show. We give a direct approach and demonstrates that such human concepts are indeed preserved, which further strengthens the work of Karras et al. [84].

In addition, four facial embedding models (*FaceNet*, *OpenFace*, *Reduced DeepFace*, and *Reduced VGG-Face*) were used to embed the images. This allowed us to cluster each combination of input and output dataset. By determining the purity score, which measures how intertwined each subcluster is, we were able to show that the generated images are not grouped together with the input dataset. This means that StyleGAN2 is able to generate new persons that do not exist in the input dataset, according to the facial embeddings. Recently, Khodadadeh et al. [85] had a similar idea of using a face recognition method in combination with StyleGAN2. They used *FaceNet* in a loss function to generate faces with StyleGAN2 that belong to the same identity. Furthermore, they used 35 attribute methods to steer the latent space in order to generate faces with modified attributes, which is different compared to our research. The insight that StyleGAN2 is capable

of generating new identities is novel and one of the contributions of our research.

Summarizing, by using a two-pronged humanly approach, consisting of predicting human attributes (Section 2.3.1) and clustering using face recognition models (Sections 2.3.2 and 2.3.3), the following conclusions can be drawn:

- The images generated by StyleGAN2 (without truncation) have the same attribute distributions as the input dataset, according to the prediction models.
- The generated faces belong with high probability to different persons compared to the input dataset, according to the clustering using face recognition models.
- Adding truncation to the latent variable space changes the attribute distributions towards the attributes corresponding with the latent variable used in truncation, according to the prediction models.

Generalizing, our approach can also be used for other face generators. It is not specifically tailored for StyleGAN2. Furthermore, our approach is modular in the sense that different attribute prediction and facial embedding methods can be added or removed. It should therefore be used in conjunction with other evaluation measures such as FID and PPL to give a broader perspective of the performance of a face generator. It addresses different questions and concerns compared to previous measures. When our approach, for example, shows that the generated images belong to identities in the input dataset, adaptations could be made if this effect is undesirable due to privacy issues.

2.5.1 Future work

Finally, we address a number of topics for future research. Section 2.4 provides multiple insights that should be explored further. First, note that the maximum number of subclusters is relatively small (see Table 2.3). Not more than 5,592 subclusters are formed maximally for a dataset consisting of 170,000 images. A lot of images are considered to be anomalies, which means that there is no face that closely resembles theirs. It could be that either the dataset is too small, due to the wide variety of possible faces, or the embedding methods are too specific.

Second, truncation ensures that the latent variables lie closer to the expected intermediate latent variable $\bar{\boldsymbol{w}}$ (see Section 2.2.1). In the results, the

attributes were very similar for the input dataset and the output dataset without truncation ($\psi = 1$). However, when truncation is added ($\psi = 0.5$), there was a shift in the attribute distributions. Our hypothesis is that this shift stems from the attribute values of the images generated with $\bar{\boldsymbol{w}}$ as intermediate latent variable (see Figure 2.2). Taking the average of the predicted attribute values for the first 1000 images, generated with $\bar{\boldsymbol{w}}$, leads to Table 2.4.

Table 2.4: Attribute predictions $\bar{\boldsymbol{w}}$: Average predicted attribute values for 1000 images (seeds 0000-0999) generated with the expected intermediate latent variable $\bar{\boldsymbol{w}}$.

Age (years)	Gender (prob.)		Race (prob.)		Horizontal rotation	Vertical rotation	Failed deepface (prob.)	Failed dlib (prob.)
23.7	Woman	0.649	Asian	0.0009	0.504	0.345	0	0
	Man	0.351	Black	0.0002				
			Middle Eastern	0.0006				
			Indian	0.0002				
			White	0.9975				
			Latino Hispanic	0.0005				

Table 2.5: Attribute predictions ($\psi = 1$): Averaged attribute values of the output dataset without truncation.

Age (years)	Gender (prob.)		Race (prob.)		Horizontal rotation	Vertical rotation	Failed deepface (prob.)	Failed dlib (prob.)
31.6	Woman	0.4158	Asian	0.2066	0.497	0.306	0.1	0.014
	Man	0.5842	Black	0.0552				
			Middle Eastern	0.0501				
			Indian	0.0100				
			White	0.5719				
			Latino Hispanic	0.1062				

When we examine the differences in the attribute distributions between truncation ($\psi = 0.5$) and no truncation ($\psi = 1$), and compare these with the difference between Tables 2.4 and 2.5, we see that they coincide. Thus, we hypothesize that adding truncation focuses the attribute distributions towards the attribute values belonging to the faces generated with the expected intermediate latent variable $\bar{\boldsymbol{w}}$. Karras et al. [84] regulate the generator to smoothen the perceptual path length of generated images under small perturbations in the latent space. This could be a reason why the human attributes are also similar under small perturbations. The hypothesis can be tested by replacing $\bar{\boldsymbol{w}}$ in the truncation procedure by a different intermediate latent variable and investigating the attribute distributions of the newly generated images. If the hypothesis holds, this method can also be used to generate images with the desired attributes. Future research is

needed to explore how $\bar{\boldsymbol{w}}$ and the truncation variable influences the attribute distributions. The attribution methods that were used are all trained on alternate datasets. It is unclear how well their performance transfers to the data that is used in this research. Nevertheless, it still provides the insight that the input and output distributions were similar. Still, it remains interesting to evaluate how well the models can transfer their learned knowledge to this dataset. Additionally, the goal of our approach was to evaluate the generator in a more humanlike way. We decided to use methods that were trained using humanly labeled data, as it was unfeasible for us to label this dataset ourselves. However, it remains uncertain how 'humanlike' these methods are. Are they actually predicting correct attribute labels for our datasets? Although this questions is beyond our scope, it is interesting to evaluate if these models are good at replacing human experts. Lastly, the facial recognition methods are trained using datasets of real images. The results showed that the generated images are embedded differently than the input images. If the facial embedding methods are also trained using generated faces, a better comparison could possibly be made.

Chapter 3

Active Pairwise Distance Learning for Efficient Labeling of Large Datasets by Human Experts

Contents

Based on *Joris Pries, Sandjai Bhulai, and Rob van der Mei* (2023): "**Active pairwise distance learning for efficient labeling of large datasets by human experts**". Accepted for publication in *Applied Intelligence*. [136]

Abstract

In many machine learning applications, the labeling of datasets is done by human experts, which is usually time-consuming in cases of large data sets. This raises the need for methods to make optimal use of the human expert by selecting model instances for which the expert opinion is of most added value. This chapter introduces the problem of *active pairwise distance learning* (APDL), where the goal is to *actively* learn the pairwise distances between *all* instances. Any distance function can be used, which means that APDL techniques can e.g., be used to determine likeness between faces or similarities between users for recommender systems. Starting with an unlabeled dataset, each round an expert determines the distance between one pair of instances. Thus, there is an important choice to make each round: 'Which combination of instances is presented to the expert?' The objective is to accurately predict all pairwise distances, while minimizing the usage of the expert. In this research, we establish upper and lower bound approximations (including an update rule) for the pairwise distances and evaluate many domain-independent query strategies. The observations from the experiments are therefore general, and the selection strategies are ideal candidates to function as baseline in future research. We show that using the criterion *max degree* consistently ranks amongst the best strategies. By using this criterion, the pairwise distances of a new dataset can be labeled much more efficiently.

3.1 Introduction

A dataset plays a critical part when solving a practical problem using machine learning (ML). Often, the goal is to predict some target variable using measured features of other variables. When gathering the data, it would be ideal if the target variable could be measured. For example, consider the task of forecasting the outside temperature using multiple other measurements, such as atmospheric pressure, wind speed and humidity. In this case, the label (temperature) can be determined efficiently. In other cases, the labels are not as easily acquired. For example, to predict if a face is visible in a photograph requires human expertise at some point to label a dataset. In such cases, human involvement is sometimes necessary, especially when a model is trained to replicate human knowledge or skills.

Labeling using a human expert is a time-consuming and costly undertaking. Therefore, efforts should be focused on maximizing the usefulness of the expert when it is too expensive to label everything. Typical questions are: 'How should the expert be deployed?' and 'Which samples should be labeled?' These questions are all part of the research field called *active learning* (AL) [155]. It is a subfield of ML dedicated to achieving the best prediction performance with as few labels as possible. To this end, a human expert can be queried about an instance each round. The expert then determines a label for this instance, which in turn can be used to update a prediction model and determine the next query. This cycle continues for a fixed number of rounds or until some other stopping criterion is met [26, 76, 181].

AL is useful in situations where simply labeling all data instances is too expensive. For example, suppose we want to label a dataset with many facial images and we are interested in learning the similarity/likeness between each combination of faces. If there are $M \in \mathbb{N}_{>0}$ faces, then there are already $\binom{M}{2} = M \cdot (M-1)/2$ pairwise combinations. To label all pairwise similarities of 1,000 faces would thus already require 499,500 comparisons. For large datasets, this quickly becomes too costly to label (either time or money wise), which is why AL techniques have been developed.

A critical aspect in AL is the selection algorithm (the so-called *query function*) that determines which samples should be given to the expert. The selection algorithm can be either pre-trained using other datasets (*transfer learning* [62, 201]), or it can be adjusted on-the-fly. AL techniques (almost always) use feature values to improve the query function, which is commonly some supervised learning method (e.g., a neural network). Yoo et al. [192] attached

a module to a target network to predict the target losses for unlabeled data. Klein et al. [90] measured anomaly scores of feature values as guidance for the query function. Another common selection criterion is some kind of uncertainty sampling [5], whereby a prediction model is trained using the labeled data, and applied on the feature values of the unlabeled data. Uncertain predictions are then queried to the expert.

In this chapter, we investigate an unexplored area within AL, that we call *active pairwise distance learning* (APDL). The objective in APDL is to actively learn the *pairwise distances* between all instances. Any distance function can be used, which means that APDL techniques can e.g., be used to determine likeness between faces or similarities between users for recommender systems. Furthermore, APDL methods can also be used in kinship recognition, deep fake detection, anomaly detection, dissimilarity sampling, and (pairwise) clustering. Studying APDL is therefore valuable for many research areas. It is important to emphasize that, we will not make any assumptions in this research about the relevance of the feature values to these distances (see Section 3.2.2 for more details), which makes our results highly generic and hence useful in many application areas.

The contribution of this research is three-fold. First, we introduce APDL, the problem of actively learning the pairwise distances between all instances. Second, we establish upper and lower bound approximations for the pairwise distances, and an update rule for these bounds. Third, we identify the best generic (domain-independent) baseline strategies for practical applications. This research can be seen as a pioneering contribution to the field of AL, which is expected to raise many follow-up studies in future research.

The remainder of this chapter is organized as follows. In Section 3.2, we formally introduce APDL and discuss why no assumptions are made about the feature values. Consequently, we argue that techniques from unsupervised learning, semi-supervised learning and reinforcement learning are not applicable without these assumptions. Related research is discussed in Section 3.3. Section 3.4 defines notation for the selection strategies. Furthermore, it is discussed how each additional pairwise distance will update the upper and lower approximation bounds for all pairwise distances. A variety of selection strategies and selection criteria are defined in Section 3.5. Next, the experimental setup is addressed in Section 3.6. The experiments evaluate the selection strategies on multiple datasets to find the best performing strategy. The results of the experiments are discussed in Section 3.7. Section 3.9 gives an extensive overview of possible future research opportunities and addresses limitations of the results presented in this chapter. Finally, Section 3.10

summarizes the findings.

3.2 Active pairwise distance learning

3.2.1 Definition of APDL

To start, we formally define *active pairwise distance learning* (APDL). Starting with an unlabeled dataset consisting of M instances, the objective of APDL is to learn as much as possible about the distance between *each pair of instances* in $T \in \mathbb{N}_{>0}$ rounds. Each round, an expert can be queried to label exactly one pairwise distance. After T rounds, a final prediction is made about all pairwise distances. Given a pre-determined loss function $\mathcal{L}$, the goal is to minimize the loss between the *actual* pairwise distance matrix $\mathcal{D}^{\text{true}}$ and the *predicted* pairwise distance matrix $\mathcal{D}^{\text{pred}}$. Thus, the target of any APDL algorithm is to minimize $\mathcal{L}\left(\mathcal{D}^{\text{pred}}, \mathcal{D}^{\text{true}}\right)$.

In general, there are two critical components in APDL: (I) '*Which pair is queried each round?*' and (II) '*How to use this information to make the best prediction?*' The first question is the main focus of this research. The general approach of an APDL algorithm can be seen in Algorithm 1.

Algorithm 1 General APDL Algorithm

Input: # samples M, # rounds T, expert labeler Ω
Output: Pairwise distance prediction $\mathcal{D}^{\text{pred}}$

1: **for** $t \leftarrow 1$ to T **do**
2: Select $(i, j) \in \{1, \ldots, M\}^2$
3: Receive distance $d(i, j)$ from expert Ω
4: **end for**
5: Make final pairwise distance prediction $\mathcal{D}^{\text{pred}}$

3.2.2 No relevancy assumption

An important assumption that we make in this research is that no assumptions are made about the relevance of the feature values to the actual distance. As a consequence, only techniques that do not use the feature values are considered. Note that having similar feature values does not necessarily mean that the underlying distance between two instances is small. Insufficient features could mean that instances appear close, but are actually far apart. Having too many features could also be troublesome for measuring similarity, as instances in a high-dimensional space are often far away (due to the infamous *curse of dimensionality*). Furthermore, sufficient

labeled data is required to accurately extract information from the feature values in order to make good predictions. Especially for high-dimensional data and complex prediction models, more labeled data is necessary to properly train the prediction model. Gal et al. [57] even identified the lack of scalability to high-dimensional data as one of the major remaining challenges for AL. However, in practice sufficient labeled data is not always available. In addition, a recent survey [140] stated that "research remains in its infancy at present, and there is still a long way to go in the future." A badly trained prediction model could steer the query selection in the wrong direction.

Without making any assumption about the relevancy of the feature values to the pairwise distance makes most known techniques from *unsupervised*, *semi-supervised* and *active learning* inappropriate. Chapelle et al. [37] identify in which cases *semi-supervised learning* is suitable. They determine the following three assumptions in order to apply semi-supervised learning techniques:

Smoothness assumption: "If two points x_1, x_2 in a high-density region are close, then so should be the corresponding outputs y_1, y_2."

Cluster assumption: "If points are in the same cluster, they are likely to be of the same class."

Manifold assumption: "The (high-dimensional) data lie (roughly) on a low-dimensional manifold."

The *smoothness* and *cluster assumption* do not have to hold when the underlying distance metric (responsible for the actual labels) is very different from the metric that is used to measure if two points are close and if they belong to the same cluster. Consider for example determining if cars are similar using images. If the distance between two images is measured by comparing them pixel-by-pixel, it is highly likely that only the color of the car determines if two cars are similar (or even the background). Therefore, this is not a good approach.

The *manifold assumption* is important to combat the well-known curse of dimensionality problem. Without this assumption, a lot of data is necessary to learn the underlying distribution from the feature values. In such a situation, it might be better to make no assumptions than being steered in the wrong direction due to a lack of labeled data.

Techniques from *reinforcement learning* [171] have similar problems, when feature values are used. Given a specific dataset, the same action (i.e.,

querying the expert about a certain pair) is not repeated. Furthermore, no state is revisited and the state space can be really large. Thus, some mapping must be learned from the feature values. This inherently has the same assumption problems as discussed before.

When not to make relevancy assumption
We identify six situations where it could be useful to make no assumptions about the relevancy of the feature values to the pairwise distance: (I) when there is not yet enough labeled data for supervised techniques; (II) when the underlying metric is unknown and could be too complex to predict using the given features; (III) when the features are not sufficient; (IV) when there are too many features; (V) when the model should work across multiple domains; (VI) as baseline to evaluate techniques that do use feature values.

To elaborate on situation (VI), whenever for example a semi-supervised technique is developed, it should perform better than any method that does not use the feature values. Therefore, not using the feature values can be used to benchmark methods that do use feature values.

Advantages of not using feature values
Not using feature values has its benefits. We list five advantages: (I) the dimensionality of data is irrelevant; (II) the quality of feature values is unimportant; (III) no hyper-parameter tuning based on feature values is needed; (IV) conclusions are not dependent on the application domain; (V) resulting baselines are ideal to be used as benchmark. As this research constitutes the first step in APDL, these are the reasons why we decide to only investigate selection strategies that do not use feature values.

3.3 Related research

To the best of our knowledge, APDL is a new research area within AL. However, there are related papers, which we will outline below.

APDL is not the same as *learning pairwise preferences* [75], where the goal is to make a ranking based on pairwise comparisons. In these pairwise comparisons, it is decided which sample is more preferable, which is a binary choice. A might be preferred over B, but it is not labeled by how much, which is an important distinction. Furthermore, the focus lies more on determining a good ranking function, not necessarily determining which samples should be labeled in order to gain the most information. However, it is closely related and (non-binary) preference / desirability could also be

used as a distance metric within APDL.

Dasarathy et al. [42] investigate binary label prediction on a graph. A non-parametric algorithm is developed to actively learn to predict binary labels in a graph. The objective for APDL is to learn *all* pairwise distances, thus the graph would be fully connected. The main difference with our research is that binary labels are assumed in [42], whereas we assume that the labels are generated by a distance metric. On the one hand, it makes the problem easier, as structure is added to the labels, because properties of a distance metric need to be satisfied. On the other hand, a label can now be real-valued and not only binary, which makes prediction much harder.

Actively learning pairwise similarities has also been studied for hierarchical clusterings [49]. The goal is to infer the hierarchical clustering using as few similarities as possible. These similarities are not necessarily from a distance metric, as e.g., the Pearson correlation is used in [49]. The performance is assessed by evaluating the constructed tree structures. This makes APDL different, as the objective is to predict all pairwise distances, not to identify the correct tree structure.

APDL is also closely related to *similarity learning* and *metric learning* [99, 184, 197]. These are supervised ML areas, where the goal is to learn from a labeled dataset a similarity function and a metric, respectively. The task of face verification is a practical example of these research areas. In [152], the triplet loss is used to learn a distance function from 0/1-labels to compare faces. The main difference with APDL is that similarity and metric learning require a labeled dataset in order to determine a generalized function that can be used for new samples. The objective in APDL is to gather as much distance-based information as possible about a fixed dataset, when there is yet no information about the labels. APDL is thus not concerned about finding a general function for samples outside the given dataset. APDL could be used to build the dataset that is later used by techniques from *similarity learning* and *metric learning*.

Metric learning has also been researched in an AL setting. Yang et al. [191] developed a Bayesian framework to actively learn a distance metric by selecting the unlabeled pairs with the greatest uncertainty in predicting whether the pair is in the same equivalence class or not. Kumaran et al. [96] actively learned a distance metric to identify outlier and boundary points per class, which are then given to the expert. Even more selection strategies are explored in [47]. Pasolli et al. [127] used an actively learned metric to reduce the dimensionality of hyperspectral images and to select

uncertain samples. Again, the goal in *active metric learning* is to get a model to accurately predict if two samples belong to the same class, not to determine an accurate prediction for pairwise distances. This makes APDL a fundamentally different problem.

3.4 Definitions and bounds

First, we introduce some notation that is necessary to discuss selection strategies. As seen in Algorithm 1, in round t a pair of indices $\zeta_t := (i, j)$ is chosen from M indices and a corresponding distance $d(i, j)$ between these indices is obtained from the expert. Although it is possible to disregard previous requests to the expert, it is obvious that previous results should be taken into account when selecting the next pair of indices. If only to avoid asking the expert the same pair twice. Therefore, we introduce the notion of *history*.

Definition 3.4.1 (History)**.** Name $\mathcal{H}_t = \{((i, j), d(i, j)) : \zeta_\tau = (i, j)\}_{\tau=1,\dots,t}$ the *history* of all chosen pairs of indices and their corresponding labeled distance up to and including round t. Furthermore, define $\mathcal{H}_0 := \varnothing \times \varnothing$.

Next, we will define what a selection strategy is. A selection strategy for T rounds consists of T functions that successively determine which pair of indices is chosen based on the given history.

Definition 3.4.2 (Selection strategy)**.** We call σ a *selection strategy* if for each $t \in \{1, \dots, T\}$ it holds that $\sigma_t : \mathcal{H}_{t-1} \mapsto \zeta_t \in \{1, \dots, M\}^2$ and $\sigma = \{\sigma_t\}_{t=1,\dots,T}$.

3.4.1 Expert distance metric

After the selection strategy determines which pair of indices is chosen, the expert determines the distance between them. An important and strong assumption we make, is that the expert makes no mistakes and that the distances originate from an underlying metric $d : \{1, \dots, M\}^2 \to [0, d_{\max}]$, where $d_{\max} \in \mathbb{R}_{>0}$ is the maximum possible distance between two samples. In most instances, $d_{\max}$ can be estimated or determined. However, when the maximum distance cannot be bounded from above, consider $d_{\max}$ to be infinite. In our experiments, the underlying distance metric is the Euclidean distance between two samples and the expert simply returns the correct Euclidean distance.

3.4.2 Approximation bounds

To approximate the true distance between each pair of indices, we can make use of the fact that the underlying distance function d is a metric, to find upper and lower bounds. Each metric satisfies, by definition, the *triangle inequality* and the subsequent *reverse triangle inequality*. Denote the upper and lower bound of (i, j) in round t as $\mathcal{D}_t^{\text{upp}}(i, j)$ and $\mathcal{D}_t^{\text{low}}(i, j)$, respectively. The metric d is symmetric (i.e., $d(x, y) = d(y, x)$), thus we enforce the upper and lower bounds to be symmetric as well. Therefore, it must always hold that $\mathcal{D}_t^{\text{upp}}(i, j) = \mathcal{D}_t^{\text{upp}}(j, i)$ and $\mathcal{D}_t^{\text{low}}(i, j) = \mathcal{D}_t^{\text{low}}(j, i)$. We will now discuss how triangle inequalities can be used to update the upper and lower bounds each time a new distance is obtained from the expert.

Initialization

In the first round, there is no distance information yet. However, as d is a metric, it must hold that $d(i, i) = 0$ for each $i \in \{1, \ldots, M\}$. Furthermore, using the range of d, the upper and lower bounds are initialized as:

$$\mathcal{D}_1^{\text{upp}}(i, j) = \begin{cases} 0 & \text{if } i = j, \\ d_{\max} & \text{else.} \end{cases}$$

$$\mathcal{D}_1^{\text{low}}(i, j) = 0.$$

Triangle inequality

The *triangle inequality* states that for all $a, b, c \in \{1, \ldots, M\}$ it must hold that $d(a, c) \leq d(a, b) + d(b, c)$. Expanding on this, for every round t it follows that

$$d(a, c) \leq d(a, b) + d(b, c) \leq \mathcal{D}_t^{\text{upp}}(a, b) + \mathcal{D}_t^{\text{upp}}(b, c).$$

In other words, $\mathcal{D}_t^{\text{upp}}(a, b) + \mathcal{D}_t^{\text{upp}}(b, c)$ is an upper bound for $d(a, c)$. Therefore, it must hold that

$$\mathcal{D}_t^{\text{upp}}(a, c) \leq \min \left\{ d_{\max}, \mathcal{D}_t^{\text{upp}}(a, b) + \mathcal{D}_t^{\text{upp}}(b, c) \right\}. \tag{3.1}$$

Reverse triangle inequality

The *reverse triangle inequality* states that $|d(a, b) - d(b, c)| \leq d(a, c)$ for all $a, b, c \in \{1, \ldots, M\}$. Now note that

$$|d(a, b) - d(b, c)| \geq \mathcal{D}_t^{\text{low}}(a, b) - \mathcal{D}_t^{\text{upp}}(b, c),$$

$$|d(a, b) - d(b, c)| \geq \mathcal{D}_t^{\text{low}}(b, c) - \mathcal{D}_t^{\text{upp}}(a, b).$$

Therefore, this gives a lower bound for (a, c). Thus,

$$\mathcal{D}_t^{\text{low}}(a,c) \geq \max\left\{0, \mathcal{D}_t^{\text{low}}(a,b) - \mathcal{D}_t^{\text{upp}}(b,c), \mathcal{D}_t^{\text{low}}(b,c) - \mathcal{D}_t^{\text{upp}}(a,b)\right\}. \tag{3.2}$$

Update rules

In round t, we first set $\mathcal{D}_{t+1}^{\text{low}} := \mathcal{D}_t^{\text{low}}, \mathcal{D}_{t+1}^{\text{upp}} := \mathcal{D}_t^{\text{upp}}$. After the new distance $d(i,j)$ is given by the expert, the upper and lower bound collapse to $d(i,j)$, as it is assumed that the expert makes no mistakes. Thus,

$$\begin{aligned} \mathcal{D}_{t+1}^{\text{low}}(i,j) &:= d(i,j) =: \mathcal{D}_{t+1}^{\text{upp}}(i,j), \\ \mathcal{D}_{t+1}^{\text{low}}(j,i) &:= d(i,j) =: \mathcal{D}_{t+1}^{\text{upp}}(j,i). \end{aligned} \tag{U1}$$

This newly acquired information can have an effect on other bounds as well. For all $k \in \{1, \ldots, M\}$ Equation (3.1) now gives the following *update rules*:

$$\begin{aligned} \mathcal{D}_{t+1}^{\text{upp}}(i,k) &:= \min\left\{d_{\max}, \mathcal{D}_{t+1}^{\text{upp}}(i,j) + \mathcal{D}_{t+1}^{\text{upp}}(j,k)\right\}, \\ \mathcal{D}_{t+1}^{\text{upp}}(j,k) &:= \min\left\{d_{\max}, \mathcal{D}_{t+1}^{\text{upp}}(i,j) + \mathcal{D}_{t+1}^{\text{upp}}(i,k)\right\}, \\ \mathcal{D}_{t+1}^{\text{upp}}(k,i) &:= \mathcal{D}_{t+1}^{\text{upp}}(i,k), \\ \mathcal{D}_{t+1}^{\text{upp}}(k,j) &:= \mathcal{D}_{t+1}^{\text{upp}}(j,k). \end{aligned} \tag{U2}$$

Note that this can lead to multiple updates, as $\mathcal{D}_{t+1}^{\text{upp}}(i,k)$ is updated in the first line and used in the second, whereas $\mathcal{D}_{t+1}^{\text{upp}}(j,k)$ is used in the first and updated in the second. For each bound that is now tighter than before, the same procedure should be repeated. Note that the order of the updates does not influence the end result as long as the effect of every tighter bound is evaluated.

Thereafter, lower bounds can be updated using Equation (3.2). The updates are as follows (for all $k \in \{1, \ldots, M\}$):

$$\begin{aligned} \mathcal{D}_{t+1}^{\text{low}}(i,k) &:= \max\left\{0, \mathcal{D}_{t+1}^{\text{low}}(i,j) - \mathcal{D}_{t+1}^{\text{upp}}(j,k), \mathcal{D}_{t+1}^{\text{low}}(j,k) - \mathcal{D}_{t+1}^{\text{upp}}(i,j)\right\}, \\ \mathcal{D}_{t+1}^{\text{low}}(j,k) &:= \max\left\{0, \mathcal{D}_{t+1}^{\text{low}}(i,j) - \mathcal{D}_{t+1}^{\text{upp}}(i,k), \mathcal{D}_{t+1}^{\text{low}}(i,k) - \mathcal{D}_{t+1}^{\text{upp}}(i,j)\right\}, \\ \mathcal{D}_{t+1}^{\text{low}}(k,i) &:= \mathcal{D}_{t+1}^{\text{low}}(i,k), \\ \mathcal{D}_{t+1}^{\text{low}}(k,j) &:= \mathcal{D}_{t+1}^{\text{low}}(j,k). \end{aligned} \tag{U3}$$

Again, this can lead to multiple updates, similar to the upper bound updates. However, it is important to note that a new upper bound can lead to a new lower bound, but not vice versa. When an upper bound changes (e.g., $\mathcal{D}^{\text{upp}}_{t+1}(x, y)$), Update rules (U2) and (U3) should be evaluated (replacing (i, j) with (x, y)). Whenever a lower bound changes (e.g., $\mathcal{D}^{\text{low}}_{t+1}(x, y)$), only Update rule (U3) needs to be checked. The entire update procedure is summarized in Algorithm 2, that should be applied each time a new distance label is obtained from the expert.

3.5 Strategies

In this section, we discuss the selection strategies that will be evaluated. As the APDL problem is new, we will investigate relatively straightforward strategies based on naturally arising criteria to determine the baseline strategies for future research. Without previous literature, there is yet no evidence which strategies should perform well. However, we can argue e.g., that selecting indices, where the upper and lower bound are already close, is not a good idea. Thus, sometimes we investigate a strategy that maximizes a criterion, without looking into a strategy that minimizes the same criterion, or vice versa. On top of the general definition of a strategy (see Definition 3.4.2), it is necessary to introduce some concepts and definitions that are used by certain selection strategies.

A selection strategy σ consists of functions σ_t for $t \in \{1, \ldots, T\}$ (see Definition 3.4.2). For all strategies that will be used, it holds that the same selection criterion is used for each σ_t. In other words, the strategy does not change for different rounds.

It is possible that multiple samples satisfy some selection criterion (for example, the *least chosen* strategy). If more than one sample is optimal for the selection criterion, a selection between these samples is made uniformly at random. The following notation is used for this.

Definition 3.5.1 (Drawn uniformly from set)**.** Let $\mathcal{U}(A)$ denote the uniform distribution over a finite non-empty set A. Thus, when $X \sim \mathcal{U}(A)$ it must hold that $\mathbb{P}(X = a) = \frac{1}{|A|}$ for each $a \in A$.

Degree

It is also useful to track how often each index is chosen. Note that the problem can be visualized by a graph. Each sample is a vertex, and an edge is drawn between a pair of vertices, whenever the expert labels the distance

Algorithm 2 Update upper and lower bounds

Input: Distance $d(x, y)$, indices (x, y), round t
Output: Updated bounds $\mathcal{D}^{\text{upp}}_{t+1}, \mathcal{D}^{\text{low}}_{t+1}$

Initialization:
$\mathcal{D}^{\text{upp}}_{t+1} \leftarrow \mathcal{D}^{\text{upp}}_{t}, \mathcal{D}^{\text{low}}_{t+1} \leftarrow \mathcal{D}^{\text{low}}_{t}$
$\mathcal{D}^{\text{low}}_{t+1}(x, y) \leftarrow d(x, y)$, $\mathcal{D}^{\text{low}}_{t+1}(y, x) \leftarrow d(x, y)$
$\mathcal{D}^{\text{upp}}_{t+1}(x, y) \leftarrow d(x, y)$, $\mathcal{D}^{\text{upp}}_{t+1}(y, x) \leftarrow d(x, y)$
$U_{\text{update}} \leftarrow \{(x, y)\}$, $L_{\text{update}} \leftarrow \{(x, y)\}$

Update upper bounds:
while $U_{\text{update}} \neq \varnothing$ **do**
    Take $(i, j) \in U_{\text{update}}$
    for $k \leftarrow 1$ to M **do**
        Update $\mathcal{D}^{\text{upp}}_{t+1}$ with Update rule (U2)
    end for
    for every tighter bound **do**
        Add corresponding indices to U_{update} and L_{update} ▷ Every tighter upper bound could lead to other new upper or lower bounds
    end for
end while
Remove duplicates from L_{update}

Update lower bounds:
while $L_{\text{update}} \neq \varnothing$ **do**
    Take $(i, j) \in L_{\text{update}}$
    for $k \leftarrow 1$ to M **do**
        Update $\mathcal{D}^{\text{low}}_{t+1}$ with Update rule (U3)
    end for
    for every tighter bound **do**
        Add corresponding indices to L_{update} ▷ Every tighter lower bound could lead to other new lower bounds
    end for
end while

between these pairs. How often each index is chosen is identical to the *degree* (from graph theory) of the corresponding vertex. Let $\deg_t(k)$ denote the *degree* of sample k in round t. This can be determined by

$$\deg_t(k) = |\{\zeta_\tau = (i,j) : i = k \vee j = k\}_{\tau=1,\dots,t-1}|.$$

Predicted distance

Let $\mathcal{D}_t^{\text{pred}}(i,j)$ be the predicted distance between samples i and j in round t. We will later show (in Definition 3.6.1 below) how the distance is actually predicted. Strategies can use these predictions in a selection criterion.

Different kinds of strategies

Next, we divide the selection strategies into two groups, namely *simultaneous* and *sequential strategies*. Behind a *simultaneous strategy*, there is a singular selection criterion that determines which pair of indices is selected in round t out of all possible remaining pairs in

$$\mathcal{I}_t := \big\{(i,j) \in \{1,\dots,M\}^2 : \sigma_\tau(\mathcal{H}_{\tau-1}) \notin \{(i,j),(j,i)\} \text{ for all } \tau \in \{1,\dots,t-1\}\big\}.$$

For a *sequential strategy*, the indices are chosen one after the other by two (possibly different) selection criteria. To this end, if $\sigma_t(\mathcal{H}_{t-1}) = (i,j)$, let $\sigma_t(\mathcal{H}_{t-1})_1 := i$ and let $\sigma_t(\mathcal{H}_{t-1})_2 := j$ denote the first and second index respectively. $\sigma_t(\mathcal{H}_{t-1})_1$ is chosen from the remaining first indices, thus from

$$\mathcal{I}_{1,t}^{\text{uniq.}} := \{i : \exists (i,\cdot) \in \mathcal{I}_t\}.$$

Whenever the first index is chosen, the remaining second indices reduce, as it is limited by the first chosen index $\sigma_t(\mathcal{H}_{t-1})_1$. The second index is chosen from

$$(\mathcal{I}_t \mid \sigma_t(\mathcal{H}_{t-1})_1)^{\text{uniq.}} := \{j : \exists (\sigma_t(\mathcal{H}_{t-1})_1, j) \in \mathcal{I}_t\}.$$

3.5.1 Simultaneous strategies

First, we will discuss the *simultaneous strategies*, where both indices are chosen at the same time.

Random pair

Select a pair uniformly at random out of the remaining pairs.

Criterion 1 (Random pair)**.**

$$\sigma_t(\mathcal{H}_{t-1}) \sim \mathcal{U}(\mathcal{I}_t). \tag{3.3}$$

Max bound gap

Select a pair uniformly at random out of the remaining pairs with the largest difference between the upper and lower bound of the predicted distance.

Criterion 2 (Max bound gap)**.**

$$\sigma_t(\mathcal{H}_{t-1}) \sim \mathcal{U}\left(\underset{(i,j)\in\mathcal{I}_t}{\arg\max}\left\{\mathcal{D}_t^{\text{upp}}(i,j) - \mathcal{D}_t^{\text{low}}(i,j)\right\}\right). \tag{3.4}$$

Max combined total bound gap

First, determine for each sample the bound gap with all other samples and sum these into a combined bound gap. Then, select a pair uniformly at random out of the remaining pairs with the largest sum of combined bound gaps.

Criterion 3 (Max combined total bound gap)**.**

$$\sigma_t(\mathcal{H}_{t-1}) \sim \mathcal{U}\left(\underset{(i,j)\in\mathcal{I}_t}{\arg\max}\left\{\sum_{k=1}^{M}\left(\mathcal{D}_t^{\text{upp}}(i,k) - \mathcal{D}_t^{\text{low}}(i,k) + \mathcal{D}_t^{\text{upp}}(j,k) - \mathcal{D}_t^{\text{low}}(j,k)\right)\right\}\right). \tag{3.5}$$

Max/min total degree

First, determine for each sample the degree, see Section 3.5. Then, select a pair uniformly at random out of all remaining pairs where the sum of the individual degrees is maximized/minimized.

Criterion 4 (Max total degree)**.**

$$\sigma_t(\mathcal{H}_{t-1}) \sim \mathcal{U}\left(\underset{(i,j)\in\mathcal{I}_t}{\arg\max}\left\{\deg_t(i) + \deg_t(j)\right\}\right). \tag{3.6}$$

Criterion 5 (Min total degree)**.**

$$\sigma_t(\mathcal{H}_{t-1}) \sim \mathcal{U}\left(\underset{(i,j)\in\mathcal{I}_t}{\arg\min}\left\{\deg_t(i)+\deg_t(j)\right\}\right). \tag{3.7}$$

3.5.2 Sequential strategies

Next, we will discuss the *sequential strategies*, where the second index is chosen after the first.

Random index

Draw uniformly at random an index out of the unique set of possible remaining indices.

Criterion 6 (Random index)**.**

$$\sigma_t(\mathcal{H}_{t-1})_1 \sim \mathcal{U}\left(\mathcal{I}_{1,t}^{\text{uniq.}}\right), \tag{3.8}$$

$$\sigma_t(\mathcal{H}_{t-1})_2 \sim \mathcal{U}\left((\mathcal{I}_t \mid \sigma_t(\mathcal{H}_{t-1})_1)^{\text{uniq.}}\right). \tag{3.9}$$

Note that choosing the first and second index using *random index* is not equivalent to using the *random pair* strategy, as *random index* uses the unique indices, where *random pair* does not.

Linked

This strategy can only be applied for the first index. Use the second index of the previous round as the first index of this round, unless there are no remaining pairs with this index. In this case and in the first round, choose the first index uniformly at random from the unique first indices, equivalent to the *random index* strategy, see Equation (3.8).

Criterion 7 (Linked)**.**

$$\sigma_t(\mathcal{H}_{t-1})_1 \sim \begin{cases} \mathcal{U}\left(\sigma_{t-1}(\mathcal{H}_{t-2})_2\right) & \text{if } t>1 \text{ and } \sigma_{t-1}(\mathcal{H}_{t-2})_2 \in \mathcal{I}_{1,t}^{\text{uniq.}}, \\ \mathcal{U}\left(\mathcal{I}_{1,t}^{\text{uniq.}}\right) & \text{else.} \end{cases} \tag{3.10}$$

Max/min degree

Choose uniformly at random an index with maximum degree (see Section 3.5) out of the unique set of possible remaining indices.

Criterion 8 (Max degree)**.**

$$\sigma_t(\mathcal{H}_{t-1})_1 \sim \mathcal{U}\left(\underset{i\in\mathcal{I}_{1,t}^{\text{uniq.}}}{\arg\max}\{\deg_t(i)\}\right), \tag{3.11}$$

$$\sigma_t(\mathcal{H}_{t-1})_2 \sim \mathcal{U}\left(\underset{j\in(\mathcal{I}_t|\sigma_t(\mathcal{H}_{t-1})_1)^{\text{uniq.}}}{\arg\max}\{\deg_t(j)\}\right). \tag{3.12}$$

Criterion 9 (Min degree)**.**

$$\sigma_t(\mathcal{H}_{t-1})_1 \sim \mathcal{U}\left(\underset{i\in\mathcal{I}_{1,t}^{\text{uniq.}}}{\arg\min}\{\deg_t(i)\}\right), \tag{3.13}$$

$$\sigma_t(\mathcal{H}_{t-1})_2 \sim \mathcal{U}\left(\underset{j\in(\mathcal{I}_t|\sigma_t(\mathcal{H}_{t-1})_1)^{\text{uniq.}}}{\arg\min}\{\deg_t(j)\}\right). \tag{3.14}$$

Max total bound gap

First, determine for each sample the bound gap with all other samples and sum these into a combined bound gap. Then, choose uniformly at random an index with maximum combined bound gap.

Criterion 10 (Max total bound gap)**.**

$$\sigma_t(\mathcal{H}_{t-1})_1 \sim \mathcal{U}\left(\underset{i\in\mathcal{I}_{1,t}^{\text{uniq.}}}{\arg\max}\left\{\sum_{k=1}^{M}\left(\mathcal{D}_t^{\text{upp}}(i,k)-\mathcal{D}_t^{\text{low}}(i,k)\right)\right\}\right), \tag{3.15}$$

$$\sigma_t(\mathcal{H}_{t-1})_2 \sim \mathcal{U}\left(\underset{j\in(\mathcal{I}_t|\sigma_t(\mathcal{H}_{t-1})_1)^{\text{uniq.}}}{\arg\max}\left\{\sum_{k=1}^{M}\left(\mathcal{D}_t^{\text{upp}}(j,k)-\mathcal{D}_t^{\text{low}}(j,k)\right)\right\}\right). \tag{3.16}$$

Max previous expected distance

In the first round, this strategy simplifies to the *random index* strategy (Section 3.5.2). Thereafter, choose uniformly at random an index out of

the unique set of the possible remaining indices, such that the predicted distance to the indices of the previous round is maximized.

Criterion 11 (Max previous expected distance)**.**

$$\sigma_t(\mathcal{H}_{t-1})_1 \sim \begin{cases} \mathcal{U}\left(\arg\max_{i\in\mathcal{I}_t}\Big\{ \right. & \\ \quad \mathcal{D}_t^{\text{pred}}(\sigma_{t-1}(\mathcal{H}_{t-2})_1, i) & \text{if } t>1, \\ \quad\quad + \mathcal{D}_t^{\text{pred}}(\sigma_{t-1}(\mathcal{H}_{t-2})_2, i)\Big\}\Big) & \\ \hline \mathcal{U}\left(\mathcal{I}_t\right) & \text{else.} \end{cases} \tag{3.17}$$

$$\sigma_t(\mathcal{H}_{t-1})_2 \sim \begin{cases} \mathcal{U}\left(\arg\max_{j\in(\mathcal{I}_t\mid\sigma_t(\mathcal{H}_{t-1})_1)^{\text{uniq.}}}\Big\{ \right. & \\ \quad \mathcal{D}_t^{\text{pred}}(\sigma_{t-1}(\mathcal{H}_{t-2})_1, j) & \text{if } t>1, \\ \quad\quad + \mathcal{D}_t^{\text{pred}}(\sigma_{t-1}(\mathcal{H}_{t-2})_2, j)\Big\}\Big) & \\ \hline \mathcal{U}\left((\mathcal{I}_t \mid \sigma_t(\mathcal{H}_{t-1})_1)^{\text{uniq.}}\right) & \text{else.} \end{cases} \tag{3.18}$$

Max/min/median expected distance

This strategy can only be applied for the second index. Select uniformly at random an index out of the unique set of remaining possible indices that belong to the maximum/minimum/median of the predicted distance (see Section 3.6.4) to the first index.

Criterion 12 (Max expected distance)**.**

$$\sigma_t(\mathcal{H}_{t-1})_2 \sim \mathcal{U}\left(\underset{j\in(\mathcal{I}_t\mid\sigma_t(\mathcal{H}_{t-1})_1)^{\text{uniq.}}}{\arg\max}\left\{\mathcal{D}_t^{\text{pred}}(\sigma_t(\mathcal{H}_{t-1})_1, j)\right\}\right). \tag{3.19}$$

Criterion 13 (Min expected distance)**.**

$$\sigma_t(\mathcal{H}_{t-1})_2 \sim \mathcal{U}\left(\underset{j\in(\mathcal{I}_t\mid\sigma_t(\mathcal{H}_{t-1})_1)^{\text{uniq.}}}{\arg\min}\left\{\mathcal{D}_t^{\text{pred}}(\sigma_t(\mathcal{H}_{t-1})_1, j)\right\}\right). \tag{3.20}$$

Criterion 14 (Median expected distance)**.**

$$\sigma_t(\mathcal{H}_{t-1})_2 \sim \mathcal{U}\left(\underset{j\in(\mathcal{I}_t\mid\sigma_t(\mathcal{H}_{t-1})_1)^{\text{uniq.}}}{\arg\text{median}}\left\{\mathcal{D}_t^{\text{pred}}(\sigma_t(\mathcal{H}_{t-1})_1, j)\right\}\right). \tag{3.21}$$

3.6 Experimental setup

3.6.1 Strategies

The goal of the experiments is to find which strategies perform well for which dataset. In Section 3.5, all used criteria are explained and defined. With *simultaneous* strategies, an index pair (i, j) is chosen at once. With *sequential* strategies, a separate decision is made for the first and second index sequentially. For example, one strategy uses Criterion 8 (max degree) to select the first index, and Criterion 9 (min degree) for the second index. In total, this leads to 5 (simultaneous) + $6 \cdot 8$ (sequential) = 53 different strategies (see Table 3.1). Furthermore, all strategies are stochastic. Therefore, each strategy is repeated ten times for each dataset. Thereafter, results are averaged to reduce stochastic outliers. It is desirable that a strategy performs generally well, not only coincidentally.

3.6.2 Data

To evaluate the performance of different strategies, fourteen two-dimensional datasets are used. To reduce computational time, the maximum allowed size of a dataset is 1,000 samples. Whenever a dataset is larger, a subset of 1,000 samples is drawn uniformly at random. The coordinates are scaled (min-max) for each dataset to be within $[0, 1]^2$. The following datasets are used, where the number of samples is denoted in round brackets: *S1* (1,000), *S2* (1,000), *S3* (1,000), *S4* (1,000) [52], *Unbalance* (1,000) [143], *Birch2-1* (1,000) [198], *Aggregation* (788) [61], *Compound* (399) [196], *Pathbased* (300), *Spiral* (312) [36], *D31* (1,000), *R15* (600) [177], *Jain* (373) [77], *Flame* (240) [56]. All these datasets are used as clustering benchmarks [53]. A visualization of these datasets can be seen in Figure 3.1. The *Euclidean distance* is used as underlying distance metric for each dataset.

Observe that these datasets are all two-dimensional. In other words, they have two features. Note that this is not a shortcoming for this experiment, as it is assumed that features are not relevant for the APDL techniques (see Section 3.2.2 above). As long as the calculated pairwise distances remain the same, these datasets could have any dimension. Two-dimensional datasets were chosen, because they can be visualized easily.

3.6.3 Number of rounds

The number of samples M is dependent on the dataset. Especially for increasingly large datasets, it is undesirable to keep on labeling until all

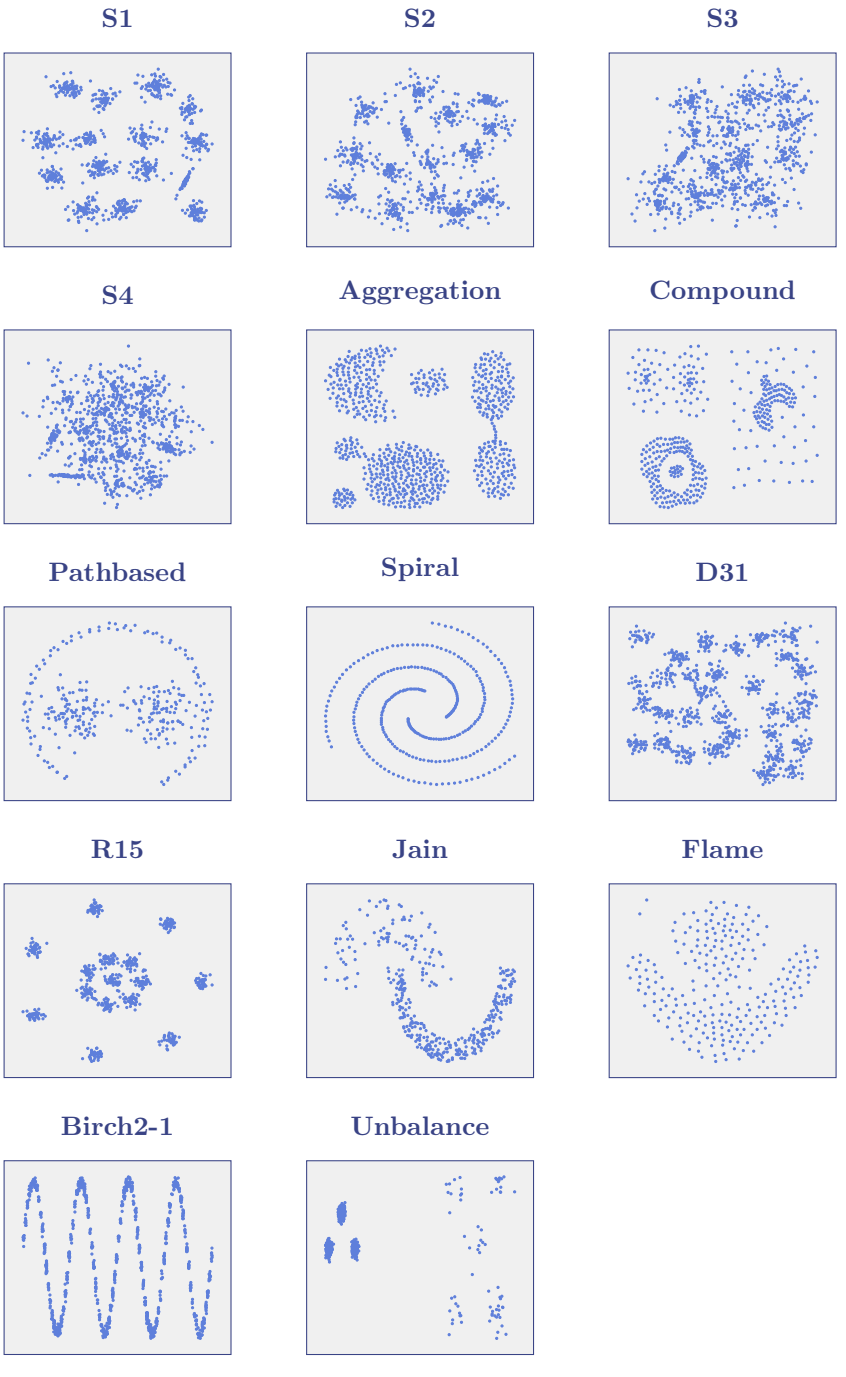


Figure 3.1: Visualization datasets: Each two-dimensional dataset that is used to test different strategies.

labels are given. Namely, $\binom{M}{2} = M \cdot (M-1)/2$ pairwise combinations can be made in total. If e.g., ten percent of the combinations should be labeled, the total number of rounds T grows exponentially in the number of samples. This gives much more opportunities to determine good upper and lower bound approximations for a large dataset compared to a small dataset. Therefore, we decide to choose the total number of rounds for a dataset in a linear-growing fashion. A *minimum spanning tree* (MST) in graph theory is a subset of edges in an undirected graph, such that all vertices are connected without any cycles. In total, $M-1$ edges are necessary to make an MST for a graph with M vertices. For each $M-1$ labels given by the expert, a minimum spanning tree could have been formed. Now, let $M_{MST} := M-1$ and define the total number of rounds T as $10 \cdot M_{MST}$. This reflects a scenario where it is not possible to determine many labels, which will often be the case in practice.

3.6.4 Performance evaluation of strategies

In order to compare the different strategies, it is important to discuss how the performance of the strategies is evaluated. Each strategy is applied ten times on each dataset. Each round a prediction is made by averaging the upper and lower bound.

Definition 3.6.1 (Predicted distance matrix)**.** Let $\mathcal{D}_t^{\text{pred}}$ be the predicted distance matrix in round t, such that

$$\mathcal{D}_t^{\text{pred}}(i,j) := \left(\mathcal{D}_t^{\text{upp}}(i,j) + \mathcal{D}_t^{\text{low}}(i,j)\right)/2.$$

Note that if (i,j) was labeled by the expert, it holds that

$$\begin{aligned}\mathcal{D}_t^{\text{pred}}(i,j) &= \left(\mathcal{D}_t^{\text{upp}}(i,j) + \mathcal{D}_t^{\text{low}}(i,j)\right)/2 = \left(d(i,j) + d(i,j)\right)/2 \\ &= d(i,j).\end{aligned}$$

Definition 3.6.2 (True distance matrix)**.** Let $\mathcal{D}^{\text{true}}$ be the true distance matrix.

The *prediction error* between the predicted distance matrix $\mathcal{D}_t^{\text{pred}}$ and the true distance matrix $\mathcal{D}^{\text{true}}$ can now be calculated. To compare these two matrices, the *mean squared error* is used. This leads to the following definition.

Definition 3.6.3 (Prediction error)**.** The error ϵ_t in round t is determined as

$$\epsilon_t = \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} \left(\mathcal{D}^{\text{true}}(i,j) - \mathcal{D}_t^{\text{pred}}(i,j) \right)^2 .$$

After collecting all prediction error results, three approaches are undertaken to compare the performance of each strategy: (I) *average performance*, (II) *Borda count*; (III) *area under the curve* (AUC). Each approach will now be explained.

Average performance

To average the prediction error results over different datasets, the error is determined at predefined rounds, specific for each dataset. As discussed in Section 3.6.3, the total number of rounds is dependent on the size of the dataset. Thus, in round $i \cdot M_{MST}$ with $i \in \{1, \ldots, 10\}$, the prediction error is determined. Averaging the results for a fixed i produces the final score. Summarizing, all prediction errors of a single strategy at predefined rounds are averaged for all ten repetitions and all fourteen datasets.

Borda count

A drawback of the previous approach is that certain datasets might be harder to predict correctly, making these datasets influence the average performance heavily, as the prediction error is relatively large, and all datasets are weighted equally. Thus, *Borda count* [17] (a voting method) is used to rank the prediction error of each strategy in the following way. First, order all strategies based on the prediction error for each dataset and repetition. The strategy with the highest prediction error gets 1 point. The second worst gets 2 points. The third highest gets 3 points and so on. This is done for each dataset and repetition in the predefined rounds $\{i \cdot M_{MST}\}_{i=1,\ldots,10}$. The final *Borda count* results are obtained by averaging over all datasets and repetitions for a fixed round. A higher score indicates better performance, and the maximum possible score is equal to the total number of strategies.

Area under the curve

Instead of comparing the results at specified iterations, it is also possible to evaluate the performance of a strategy by measuring the so-called *area*

under the curve (AUC) for each iteration using the trapezoidal rule. For each strategy, dataset and repetition, the area under the prediction error is measured up to and including the maximum number of rounds ($10 \cdot M_{MST}$). As the rounds are equally spaced, AUC reduces to

$$\sum_{t=2}^{10 \cdot M_{MST}-1} \epsilon_t + \frac{\epsilon_1 + \epsilon_{10 \cdot M_{MST}}}{2}. \tag{3.22}$$

Note that the AUC is not necessarily bounded by [0,1]. By averaging over the repetitions, an average AUC score can be derived for each strategy and dataset. A lower score indicates better performance, as the prediction error must be minimized and the sooner this is achieved the better. A fictitious example of how the AUC is measured can be seen in Figure 3.2.

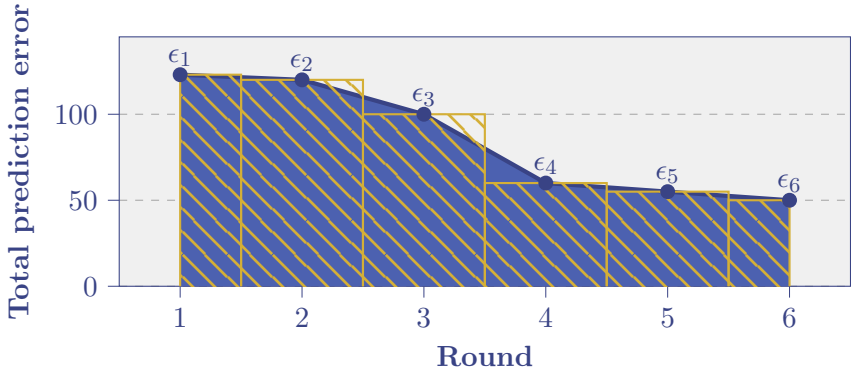


Figure 3.2: Example AUC: Each round the prediction error is measured. The AUC of the prediction error is then determined by using the trapezoidal rule (Equation (3.22)), which adds the area of the golden rectangles. Note that this is exactly equal to the blue area under the prediction error curve.

3.7 Results

Following the experimental setup from Section 3.6, all 53 strategies outlined in Section 3.5 are evaluated on fourteen different datasets (see Section 3.6.2). The results are summarized into three tables: Table 3.1 gives the average prediction error results (Section 3.6.4); Table 3.2 shows the average *Borda count* score for each strategy (Section 3.6.4); Table 3.3 displays the *area under the curve* results for each strategy and dataset (Section 3.6.4). These tables all provide a different angle on the performance of the selection strategies.

Table 3.1: Average performance: For each dataset and repetition, the prediction error of a strategy is averaged in rounds $i \cdot M_{MST}$ with $i \in \{1, \ldots, 10\}$. The ranking (by column) of each average prediction error is noted in brackets. Coloring of each column is done linearly between the worst and baseline (random pair) score and linearly between the baseline (random pair) and the best score.

Strategy	1 M_{MST}	2 M_{MST}	3 M_{MST}	4 M_{MST}	5 M_{MST}	6 M_{MST}	7 M_{MST}	8 M_{MST}	9 M_{MST}	10 M_{MST}
max degree/max degree	0.040 (02)	0.024 (07)	0.017 (10)	0.012 (10)	0.009 (09)	0.006 (09)	0.005 (12)	0.004 (12)	0.003 (13)	0.003 (13)
max degree/min degree	0.041 (09)	0.024 (09)	0.016 (07)	0.011 (07)	0.008 (06)	0.006 (06)	0.005 (08)	0.004 (18)	0.003 (15)	0.003 (17)
max degree/max exp. dist.	0.040 (03)	0.022 (04)	0.013 (04)	0.009 (04)	0.007 (04)	0.005 (04)	0.004 (05)	0.003 (11)	0.002 (11)	0.002 (11)
max degree/max prev. exp. dist.	0.041 (07)	0.022 (05)	0.014 (06)	0.010 (05)	0.007 (05)	0.006 (05)	0.005 (09)	0.004 (13)	0.003 (14)	0.003 (15)
max degree/max total bound gap	0.041 (08)	0.022 (03)	0.013 (02)	0.008 (02)	0.005 (02)	0.004 (03)	0.003 (03)	0.002 (10)	0.002 (09)	0.001 (09)
max degree/median exp. dist.	0.042 (11)	0.023 (06)	0.014 (05)	0.010 (06)	0.008 (07)	0.006 (07)	0.005 (11)	0.004 (16)	0.003 (17)	0.003 (16)
max degree/min exp. dist.	**0.039** (01)	0.032 (13)	0.024 (13)	0.020 (13)	0.017 (14)	0.015 (21)	0.012 (21)	0.010 (21)	0.009 (22)	0.008 (23)
max degree/random index	0.042 (10)	0.025 (12)	0.017 (09)	0.012 (09)	0.009 (11)	0.007 (12)	0.005 (17)	0.005 (19)	0.004 (18)	0.003 (19)
linked/max degree	0.056 (14)	0.024 (08)	0.021 (12)	0.011 (08)	0.010 (12)	0.006 (08)	0.006 (18)	0.004 (17)	0.004 (19)	0.003 (18)
linked/min degree	0.067 (44)	0.054 (29)	0.045 (30)	0.035 (30)	0.026 (31)	0.019 (30)	0.015 (29)	0.012 (29)	0.010 (28)	0.008 (29)
linked/max exp. dist.	0.063 (16)	0.053 (16)	0.043 (17)	0.031 (16)	0.020 (16)	0.012 (15)	0.004 (06)	**0.001** (03)	**0.000** (02)	**0.000** (03)
linked/max prev. exp. dist.	0.063 (21)	0.053 (19)	0.044 (24)	0.033 (25)	0.022 (20)	0.013 (18)	0.005 (14)	0.002 (06)	0.001 (06)	**0.000** (06)
linked/max total bound gap	0.067 (42)	0.056 (36)	0.050 (42)	0.041 (39)	0.032 (39)	0.024 (35)	0.018 (33)	0.014 (33)	0.011 (31)	0.009 (31)
linked/median exp. dist.	0.063 (18)	0.058 (48)	0.050 (39)	0.042 (44)	0.036 (48)	0.033 (48)	0.030 (48)	0.029 (48)	0.028 (48)	0.027 (48)
linked/min exp. dist.	0.066 (29)	0.065 (49)	0.063 (49)	0.061 (49)	0.060 (49)	0.059 (49)	0.058 (49)	0.057 (49)	0.057 (49)	0.056 (49)
linked/random index	0.062 (15)	0.053 (18)	0.042 (16)	0.033 (20)	0.025 (26)	0.019 (24)	0.015 (27)	0.012 (30)	0.010 (30)	0.008 (30)
min degree/max degree	0.040 (04)	0.025 (11)	0.018 (11)	0.012 (12)	0.009 (10)	0.007 (11)	0.005 (15)	0.004 (15)	0.003 (12)	0.003 (12)
min degree/min degree	0.067 (43)	0.054 (27)	0.045 (29)	0.034 (29)	0.026 (29)	0.019 (29)	0.015 (26)	0.012 (27)	0.010 (29)	0.008 (27)
min degree/max exp. dist.	0.066 (35)	0.055 (33)	0.044 (23)	0.031 (17)	0.021 (17)	0.011 (14)	0.004 (07)	**0.001** (05)	0.001 (07)	**0.000** (07)
min degree/max prev. exp. dist.	0.066 (33)	0.056 (40)	0.048 (35)	0.038 (35)	0.030 (34)	0.024 (36)	0.019 (40)	0.016 (42)	0.013 (43)	0.011 (43)
min degree/max total bound gap	0.067 (47)	0.057 (42)	0.050 (43)	0.042 (43)	0.033 (42)	0.026 (45)	0.020 (43)	0.016 (41)	0.013 (41)	0.010 (39)
min degree/median exp. dist.	0.066 (31)	0.054 (22)	0.044 (21)	0.033 (23)	0.025 (23)	0.019 (26)	0.016 (30)	0.013 (31)	0.012 (38)	0.010 (41)
min degree/min exp. dist.	0.067 (53)	0.067 (53)	0.066 (53)	0.066 (53)	0.066 (53)	0.065 (53)	0.065 (53)	0.065 (53)	0.064 (53)	0.064 (53)
min degree/random index	0.066 (30)	0.054 (24)	0.045 (27)	0.034 (28)	0.026 (30)	0.020 (31)	0.015 (28)	0.012 (28)	0.010 (25)	0.008 (25)
max prev. exp. dist./max degree	0.041 (05)	0.022 (02)	0.013 (03)	0.008 (03)	0.005 (03)	0.004 (02)	0.003 (02)	0.002 (09)	0.002 (10)	0.002 (10)
max prev. exp. dist./min degree	0.066 (34)	0.056 (41)	0.048 (34)	0.038 (36)	0.031 (35)	0.024 (38)	0.020 (41)	0.016 (44)	0.013 (44)	0.011 (44)
max prev. exp. dist./max exp. dist.	0.064 (27)	0.054 (25)	0.045 (25)	0.033 (24)	0.022 (19)	0.013 (19)	0.005 (13)	**0.001** (04)	**0.000** (03)	**0.000** (02)
max prev. exp. dist./max prev. exp. dist.	0.064 (28)	0.056 (34)	0.048 (36)	0.039 (37)	0.031 (36)	0.023 (32)	0.016 (32)	0.010 (22)	0.007 (21)	0.005 (20)
max prev. exp. dist./max total bound gap	0.067 (39)	0.058 (46)	0.052 (48)	0.043 (47)	0.035 (46)	0.027 (46)	0.021 (46)	0.016 (45)	0.013 (45)	0.011 (45)
max prev. exp. dist./median exp. dist.	0.064 (26)	0.055 (31)	0.047 (31)	0.038 (34)	0.031 (37)	0.025 (43)	0.022 (47)	0.019 (47)	0.017 (47)	0.015 (47)
max prev. exp. dist./min exp. dist.	0.067 (51)	0.067 (52)	0.066 (52)	0.066 (52)	0.065 (52)	0.065 (52)	0.064 (51)	0.064 (51)	0.064 (51)	0.063 (51)
max prev. exp. dist./random index	0.064 (23)	0.055 (30)	0.047 (33)	0.038 (33)	0.030 (33)	0.023 (34)	0.018 (35)	0.014 (36)	0.011 (34)	0.009 (34)
max total bound gap/max degree	0.041 (06)	**0.017** (01)	**0.008** (01)	**0.005** (01)	**0.004** (01)	**0.003** (01)	**0.002** (01)	0.002 (08)	0.002 (08)	0.001 (08)
max total bound gap/min degree	0.067 (46)	0.057 (43)	0.050 (44)	0.042 (42)	0.033 (43)	0.025 (44)	0.020 (42)	0.016 (40)	0.013 (40)	0.010 (38)
max total bound gap/max exp. dist.	0.067 (38)	0.056 (35)	0.049 (37)	0.037 (31)	0.024 (21)	0.014 (20)	0.006 (19)	0.002 (07)	0.001 (05)	**0.000** (05)
max total bound gap/max prev. exp. dist.	0.067 (41)	0.058 (45)	0.052 (47)	0.043 (48)	0.035 (47)	0.027 (47)	0.020 (45)	0.016 (43)	0.013 (42)	0.010 (40)
max total bound gap/max total bound gap	0.067 (49)	0.057 (44)	0.051 (45)	0.043 (45)	0.033 (44)	0.025 (41)	0.019 (38)	0.015 (38)	0.011 (36)	0.009 (36)
max total bound gap/median exp. dist.	0.067 (36)	0.056 (37)	0.050 (38)	0.041 (38)	0.032 (40)	0.025 (40)	0.020 (44)	0.017 (46)	0.015 (46)	0.014 (46)
max total bound gap/min exp. dist.	0.067 (48)	0.066 (50)	0.066 (50)	0.066 (50)	0.065 (50)	0.065 (50)	0.064 (50)	0.064 (50)	0.063 (50)	0.063 (50)
max total bound gap/random index	0.067 (40)	0.056 (38)	0.050 (41)	0.041 (40)	0.032 (38)	0.024 (37)	0.018 (36)	0.014 (35)	0.011 (32)	0.009 (32)
random index/max degree	0.055 (13)	0.041 (14)	0.029 (14)	0.021 (14)	0.016 (13)	0.012 (16)	0.009 (20)	0.007 (20)	0.006 (20)	0.005 (21)
random index/min degree	0.066 (32)	0.054 (23)	0.045 (26)	0.034 (26)	0.025 (27)	0.019 (25)	0.015 (24)	0.012 (23)	0.009 (23)	0.008 (22)
random index/max exp. dist.	0.064 (22)	0.053 (20)	0.042 (15)	0.030 (15)	0.019 (15)	0.010 (13)	0.003 (04)	**0.001** (01)	**0.000** (04)	**0.000** (04)
random index/max prev. exp. dist.	0.064 (24)	0.055 (32)	0.047 (32)	0.037 (32)	0.029 (32)	0.023 (33)	0.018 (34)	0.014 (34)	0.011 (33)	0.009 (33)
random index/max total bound gap	0.067 (37)	0.056 (39)	0.050 (40)	0.041 (41)	0.032 (41)	0.024 (39)	0.019 (37)	0.014 (37)	0.011 (35)	0.009 (35)
random index/median exp. dist.	0.063 (19)	0.053 (21)	0.043 (19)	0.032 (18)	0.025 (22)	0.019 (27)	0.016 (31)	0.013 (32)	0.012 (39)	0.010 (42)
random index/min exp. dist.	0.067 (52)	0.067 (51)	0.066 (51)	0.066 (51)	0.065 (51)	0.065 (51)	0.064 (52)	0.064 (52)	0.064 (52)	0.063 (52)
random index/random index	0.063 (20)	0.053 (15)	0.043 (18)	0.033 (21)	0.025 (24)	0.019 (23)	0.015 (23)	0.012 (24)	0.010 (27)	0.008 (28)
max bound gap	0.064 (25)	0.054 (26)	0.044 (22)	0.032 (19)	0.021 (18)	0.012 (17)	0.005 (10)	**0.001** (02)	**0.000** (01)	**0.000** (01)
max total degree	0.043 (12)	0.025 (10)	0.016 (08)	0.012 (11)	0.009 (08)	0.006 (10)	0.005 (16)	0.004 (14)	0.003 (16)	0.003 (14)
min total degree	0.067 (45)	0.054 (28)	0.045 (28)	0.034 (27)	0.026 (28)	0.019 (28)	0.015 (25)	0.012 (26)	0.010 (26)	0.008 (24)
max combined total bound gap	0.067 (50)	0.058 (47)	0.051 (46)	0.043 (46)	0.034 (45)	0.025 (42)	0.019 (39)	0.015 (39)	0.011 (37)	0.009 (37)
random pair	0.063 (17)	0.053 (17)	0.043 (20)	0.033 (22)	0.025 (25)	0.019 (22)	0.015 (22)	0.012 (25)	0.010 (24)	0.008 (26)

Coloring by column:

Worst strategy — Random pair — Best strategy

Table 3.2: Borda count: For each dataset and repetition, Borda count is used to rank the prediction error of the strategies and averaged in rounds $i \cdot M_{MST}$ with $i \in \{1, \ldots, 10\}$. The ranking (by column) of each Borda count score is noted in brackets. Coloring of each column is done linearly between the worst and baseline (random pair) score and linearly between the baseline (random pair) and the best score.

Strategy	1 M_{MST}	2 M_{MST}	3 M_{MST}	4 M_{MST}	5 M_{MST}	6 M_{MST}	7 M_{MST}	8 M_{MST}	9 M_{MST}	10 M_{MST}
max degree/max degree	43.46 (02)	46.62 (06)	46.31 (07)	45.19 (08)	43.91 (08)	42.75 (09)	40.91 (14)	38.84 (14)	38.37 (14)	38.11 (14)
max degree/min degree	40.66 (13)	46.34 (07)	46.10 (10)	45.59 (06)	45.17 (05)	43.00 (06)	40.57 (15)	38.19 (18)	37.83 (15)	37.11 (19)
max degree/max exp. dist.	**43.51** (01)	47.64 (02)	48.22 (02)	47.34 (04)	46.90 (04)	45.37 (04)	43.78 (09)	41.74 (11)	41.36 (11)	41.06 (11)
max degree/max prev. exp. dist.	40.16 (14)	46.31 (08)	47.04 (05)	45.94 (05)	44.96 (06)	42.97 (07)	41.30 (12)	39.01 (12)	38.79 (12)	38.28 (13)
max degree/max total bound gap	42.29 (11)	47.13 (04)	47.68 (04)	47.52 (03)	48.19 (03)	47.06 (03)	45.74 (06)	43.69 (10)	43.11 (09)	43.26 (09)
max degree/median exp. dist.	42.34 (09)	46.69 (05)	46.94 (06)	45.37 (07)	43.93 (07)	42.90 (08)	40.94 (13)	38.58 (16)	37.78 (16)	37.99 (15)
max degree/min exp. dist.	43.24 (03)	41.34 (13)	40.09 (13)	37.11 (15)	34.78 (19)	31.71 (21)	29.17 (21)	27.84 (21)	27.58 (22)	26.54 (26)
max degree/random index	42.99 (07)	45.83 (12)	46.21 (08)	44.57 (11)	43.78 (09)	41.44 (14)	39.87 (18)	37.56 (19)	37.66 (18)	37.47 (18)
linked/max degree	42.58 (08)	46.14 (09)	43.04 (12)	44.79 (10)	41.71 (12)	42.57 (10)	39.32 (19)	38.39 (17)	37.01 (19)	37.48 (17)
linked/min degree	13.89 (45)	20.79 (34)	24.35 (34)	27.42 (31)	27.72 (31)	27.12 (29)	26.93 (28)	27.07 (25)	26.97 (26)	26.51 (27)
linked/max exp. dist.	38.21 (16)	35.71 (16)	34.23 (16)	35.76 (16)	38.23 (14)	42.17 (11)	47.11 (03)	50.71 (02)	51.16 (03)	51.29 (03)
linked/max prev. exp. dist.	37.60 (18)	34.09 (18)	29.05 (23)	30.17 (24)	34.40 (20)	39.13 (18)	43.35 (10)	46.46 (06)	47.74 (06)	47.81 (06)
linked/max total bound gap	14.34 (41)	12.41 (44)	11.38 (44)	12.71 (42)	15.25 (39)	17.66 (35)	19.81 (33)	21.41 (31)	22.93 (31)	24.12 (31)
linked/median exp. dist.	37.86 (17)	34.88 (17)	33.81 (19)	28.28 (27)	22.55 (32)	17.09 (38)	11.54 (47)	07.94 (48)	06.32 (48)	06.14 (48)
linked/min exp. dist.	18.53 (35)	11.21 (47)	07.69 (49)	05.49 (49)	04.35 (49)	03.73 (49)	03.59 (50)	03.49 (50)	03.39 (50)	03.38 (51)
linked/random index	38.90 (15)	36.02 (15)	35.28 (15)	32.00 (20)	29.36 (26)	27.19 (28)	26.11 (29)	26.05 (29)	25.69 (30)	25.42 (30)
min degree/max degree	43.01 (06)	45.94 (11)	45.27 (11)	44.14 (12)	43.70 (11)	41.82 (13)	40.36 (17)	38.96 (13)	38.56 (13)	38.40 (12)
min degree/min degree	14.14 (42)	21.27 (33)	24.84 (31)	27.71 (29)	27.88 (29)	27.81 (26)	27.34 (24)	27.03 (26)	26.72 (27)	26.82 (25)
min degree/max exp. dist.	21.81 (29)	23.04 (31)	27.22 (26)	33.44 (17)	36.76 (15)	39.98 (17)	44.06 (08)	46.84 (05)	46.91 (07)	46.43 (07)
min degree/max prev. exp. dist.	20.96 (34)	20.28 (37)	20.19 (37)	19.99 (36)	18.81 (35)	17.20 (37)	16.35 (39)	16.24 (41)	16.46 (41)	16.61 (39)
min degree/max total bound gap	13.29 (47)	12.62 (43)	12.29 (43)	12.34 (44)	12.50 (43)	12.94 (45)	13.34 (42)	14.14 (43)	14.99 (43)	15.60 (41)
min degree/median exp. dist.	21.01 (33)	23.68 (29)	28.20 (24)	31.29 (22)	29.77 (24)	26.88 (30)	23.87 (30)	20.34 (32)	17.09 (39)	14.72 (42)
min degree/min exp. dist.	05.39 (53)	01.86 (53)	01.61 (53)	01.51 (53)	01.49 (53)	01.40 (53)	01.43 (53)	01.44 (53)	01.45 (53)	01.47 (53)
min degree/random index	21.49 (30)	23.05 (30)	25.66 (29)	27.71 (29)	27.74 (30)	27.25 (27)	27.19 (26)	27.28 (24)	27.21 (25)	26.90 (23)
max prev. exp. dist./max degree	43.21 (04)	47.47 (03)	47.84 (03)	48.09 (02)	48.55 (02)	47.88 (02)	46.40 (04)	43.76 (09)	42.84 (10)	42.66 (10)
max prev. exp. dist./min degree	21.46 (31)	20.29 (36)	20.30 (35)	19.91 (37)	18.75 (37)	17.57 (36)	17.04 (38)	17.01 (40)	17.38 (38)	17.51 (38)
max prev. exp. dist./max exp. dist.	31.54 (24)	29.64 (24)	27.73 (25)	29.32 (25)	34.91 (18)	40.27 (16)	45.31 (07)	49.99 (04)	51.44 (02)	51.74 (02)
max prev. exp. dist./max prev. exp. dist.	30.52 (27)	26.53 (28)	20.22 (36)	17.09 (38)	16.27 (38)	17.99 (33)	22.00 (32)	26.03 (30)	28.09 (21)	29.36 (21)
max prev. exp. dist./max total bound gap	14.62 (39)	11.40 (45)	10.76 (46)	10.26 (46)	10.16 (48)	10.49 (48)	11.24 (48)	11.81 (45)	12.46 (45)	12.64 (45)
max prev. exp. dist./median exp. dist.	30.02 (28)	28.54 (26)	25.76 (28)	22.29 (32)	18.78 (36)	15.84 (41)	12.90 (44)	10.39 (46)	08.44 (47)	07.79 (47)
max prev. exp. dist./min exp. dist.	07.80 (51)	02.94 (52)	02.90 (52)	02.96 (52)	03.06 (52)	03.31 (51)	03.30 (51)	03.38 (51)	03.51 (49)	03.50 (49)
max prev. exp. dist./random index	31.28 (25)	28.76 (25)	24.59 (32)	20.96 (34)	18.87 (34)	17.81 (34)	17.79 (36)	18.49 (37)	19.06 (37)	20.40 (37)
max total bound gap/max degree	42.34 (09)	**49.59** (01)	**51.37** (01)	**51.10** (01)	**50.51** (01)	**49.70** (01)	47.79 (02)	45.59 (08)	44.06 (08)	43.91 (08)
max total bound gap/min degree	13.41 (46)	12.66 (42)	12.32 (42)	12.53 (43)	12.35 (44)	12.71 (46)	13.31 (43)	14.29 (42)	15.00 (42)	15.78 (40)
max total bound gap/max exp. dist.	14.11 (43)	13.06 (39)	14.75 (38)	21.00 (33)	30.49 (21)	37.36 (19)	42.57 (11)	46.46 (06)	48.69 (05)	49.04 (05)
max total bound gap/max prev. exp. dist.	14.58 (40)	11.34 (46)	11.01 (45)	10.49 (45)	10.36 (47)	10.91 (47)	11.58 (46)	12.45 (44)	13.26 (44)	13.55 (44)
max total bound gap/max total bound gap	09.29 (49)	07.44 (48)	08.31 (47)	09.25 (47)	11.72 (45)	13.91 (42)	15.70 (41)	17.39 (39)	19.23 (35)	20.52 (35)
max total bound gap/median exp. dist.	16.00 (36)	13.35 (38)	14.35 (39)	14.64 (39)	14.33 (41)	13.51 (44)	11.68 (45)	09.90 (47)	08.80 (46)	08.23 (46)
max total bound gap/min exp. dist.	12.62 (48)	04.34 (50)	04.16 (50)	03.98 (50)	03.84 (50)	03.72 (50)	03.61 (49)	03.51 (49)	03.34 (52)	03.21 (52)
max total bound gap/random index	15.07 (38)	13.03 (40)	12.48 (40)	13.14 (40)	14.75 (40)	17.09 (39)	18.56 (35)	19.94 (34)	21.00 (32)	21.87 (32)
random index/max degree	41.01 (12)	38.94 (14)	39.03 (14)	37.55 (14)	36.11 (16)	35.06 (20)	33.71 (20)	33.06 (20)	33.14 (20)	33.04 (20)
random index/min degree	21.44 (32)	23.00 (32)	26.15 (27)	28.52 (26)	28.31 (28)	27.96 (23)	27.30 (25)	27.46 (22)	27.58 (22)	27.12 (22)
random index/max exp. dist.	31.72 (21)	32.44 (19)	34.10 (17)	37.59 (13)	40.10 (13)	43.56 (05)	**47.88** (01)	50.64 (03)	50.46 (04)	49.96 (04)
random index/max prev. exp. dist.	30.94 (26)	28.54 (26)	24.94 (30)	20.72 (35)	19.06 (33)	18.68 (32)	18.89 (34)	19.34 (35)	20.33 (33)	21.24 (33)
random index/max total bound gap	15.25 (37)	13.02 (41)	12.45 (41)	12.71 (41)	13.61 (42)	15.94 (40)	17.47 (37)	18.74 (36)	19.73 (34)	20.91 (34)
random index/median exp. dist.	31.89 (20)	32.00 (22)	33.86 (18)	33.01 (18)	29.82 (23)	26.51 (31)	23.29 (31)	20.17 (33)	16.83 (40)	14.17 (43)
random index/min exp. dist.	07.36 (52)	03.06 (51)	03.07 (51)	03.09 (51)	03.09 (51)	03.12 (52)	03.19 (52)	03.27 (52)	03.38 (51)	03.49 (50)
random index/random index	32.49 (19)	32.31 (20)	33.79 (20)	32.31 (19)	29.74 (25)	27.91 (24)	27.07 (27)	26.89 (28)	26.67 (28)	26.26 (29)
max bound gap	31.59 (23)	30.11 (23)	29.77 (22)	31.09 (23)	36.02 (17)	40.90 (15)	46.05 (05)	**51.18** (01)	**52.54** (01)	**52.64** (01)
max total degree	43.04 (05)	46.07 (10)	46.17 (09)	44.92 (09)	43.71 (10)	42.01 (12)	40.38 (16)	38.59 (15)	37.70 (17)	37.83 (16)
min total degree	13.90 (44)	20.76 (35)	24.48 (33)	28.05 (28)	28.33 (27)	27.84 (25)	27.66 (22)	27.39 (23)	27.25 (24)	26.86 (24)
max combined total bound gap	09.19 (50)	07.39 (49)	08.14 (48)	09.16 (48)	11.67 (46)	13.89 (43)	15.91 (40)	17.75 (38)	19.19 (36)	20.51 (36)
random pair	31.65 (22)	32.10 (21)	33.49 (21)	31.90 (21)	29.86 (22)	28.38 (22)	27.44 (23)	26.91 (27)	26.55 (29)	26.30 (28)

Coloring by column:

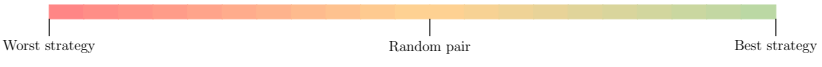


Table 3.3: AUC: For each repetition, the area under the curve (AUC) of the prediction error for a strategy is measured and averaged in rounds $i \cdot M_{MST}$ with $i \in \{1, \ldots, 10\}$. The ranking (by column) of each AUC score is noted in brackets. Coloring of each column is done linearly between the worst and baseline (random pair) score and linearly between the baseline (random pair) and the best score.

Strategy	aggregation	Birch2-1	compound	D31	flame	jain	pathbased	R15	s1	s2	s3	s4	spiral	Unbalance
max degree/max degree	172.7 (12)	144.3 (06)	046.0 (03)	234.6 (07)	010.6 (08)	055.0 (10)	053.7 (11)	086.4 (04)	204.1 (06)	188.2 (06)	218.4 (12)	144.0 (09)	058.3 (12)	123.5 (04)
max degree/min degree	157.2 (07)	158.2 (10)	056.1 (09)	242.0 (08)	011.0 (10)	048.5 (07)	046.4 (06)	107.7 (12)	208.1 (07)	200.9 (12)	168.8 (03)	151.0 (11)	054.5 (06)	123.9 (06)
max degree/max exp. dist.	134.3 (04)	104.9 (02)	048.5 (05)	201.6 (04)	008.2 (02)	044.2 (04)	040.7 (02)	086.0 (03)	178.1 (04)	153.2 (04)	178.3 (05)	120.9 (03)	048.1 (04)	120.4 (02)
max degree/max prev. exp. dist.	157.0 (06)	115.4 (04)	049.0 (06)	216.4 (05)	010.9 (09)	047.3 (06)	044.3 (05)	091.4 (06)	177.4 (02)	162.4 (05)	174.8 (04)	133.9 (07)	049.1 (05)	123.7 (05)
max degree/max total bound gap	124.1 (03)	107.1 (03)	046.3 (04)	188.6 (02)	008.6 (03)	043.2 (02)	040.8 (03)	088.2 (05)	181.3 (05)	150.5 (03)	162.4 (02)	125.7 (05)	045.1 (02)	120.6 (03)
max degree/median exp. dist.	154.3 (05)	148.2 (08)	053.1 (07)	219.1 (06)	011.3 (11)	045.8 (05)	047.8 (08)	105.0 (10)	210.2 (08)	188.4 (07)	191.7 (08)	144.7 (10)	054.8 (08)	157.5 (11)
max degree/min exp. dist.	220.9 (13)	175.5 (13)	075.4 (13)	344.9 (14)	015.9 (21)	073.7 (14)	089.5 (34)	140.3 (15)	318.1 (14)	286.1 (14)	267.2 (13)	186.8 (13)	072.7 (14)	131.5 (08)
max degree/random index	168.7 (11)	164.4 (12)	057.0 (10)	256.6 (12)	010.1 (06)	053.9 (09)	052.5 (10)	097.7 (07)	219.2 (10)	189.5 (08)	213.8 (10)	132.3 (06)	055.3 (09)	141.3 (10)
linked/max degree	160.2 (09)	156.8 (09)	067.6 (12)	251.1 (11)	011.4 (12)	058.5 (11)	049.9 (09)	102.9 (09)	226.2 (12)	199.1 (11)	217.5 (11)	152.8 (12)	054.8 (07)	234.9 (16)
linked/min degree	298.5 (30)	273.9 (26)	098.9 (26)	489.7 (30)	017.8 (29)	093.6 (28)	083.0 (25)	188.2 (28)	426.8 (27)	370.7 (30)	443.1 (31)	321.6 (29)	099.6 (32)	238.8 (18)
linked/max exp. dist.	238.5 (16)	209.0 (15)	082.3 (16)	408.3 (16)	013.6 (14)	075.6 (16)	063.6 (13)	138.3 (14)	344.6 (17)	298.5 (17)	388.3 (16)	289.0 (16)	075.0 (17)	232.3 (15)
linked/max prev. exp. dist.	250.5 (20)	218.9 (19)	084.2 (18)	421.2 (20)	014.3 (19)	078.3 (20)	069.3 (18)	145.6 (18)	354.6 (18)	312.8 (19)	398.8 (19)	297.0 (18)	077.7 (20)	238.3 (17)
linked/max total bound gap	329.2 (42)	301.8 (37)	109.5 (38)	527.9 (41)	018.7 (39)	104.0 (37)	092.1 (35)	213.2 (36)	467.0 (44)	405.0 (42)	495.2 (42)	362.2 (44)	103.0 (38)	283.1 (31)
linked/median exp. dist.	314.5 (35)	400.5 (48)	137.0 (48)	489.9 (31)	019.8 (48)	121.5 (48)	095.2 (41)	209.0 (34)	431.8 (33)	374.6 (32)	465.1 (33)	330.6 (32)	102.6 (36)	1176.5 (48)
linked/min exp. dist.	474.1 (49)	594.0 (49)	219.9 (49)	718.0 (49)	029.0 (49)	191.7 (49)	158.1 (52)	311.9 (49)	616.2 (49)	540.3 (49)	750.0 (49)	535.5 (49)	145.8 (49)	1445.3 (49)
linked/random index	288.5 (22)	263.2 (22)	095.2 (22)	470.0 (22)	017.5 (23)	091.4 (22)	082.7 (24)	179.7 (22)	413.2 (22)	358.0 (22)	419.9 (21)	305.7 (21)	096.2 (22)	277.2 (30)
min degree/max degree	166.3 (10)	162.1 (11)	053.2 (08)	250.3 (10)	010.0 (05)	059.8 (12)	047.5 (07)	106.0 (11)	210.6 (09)	189.8 (09)	212.5 (09)	137.5 (08)	057.8 (11)	124.8 (07)
min degree/min degree	296.5 (27)	277.4 (30)	099.4 (29)	490.6 (32)	017.8 (25)	094.3 (31)	084.6 (29)	188.7 (29)	427.5 (29)	374.0 (31)	439.6 (29)	318.3 (27)	098.5 (30)	251.6 (26)
min degree/max exp. dist.	246.7 (17)	217.0 (17)	084.6 (20)	413.4 (17)	014.0 (16)	077.5 (17)	070.3 (19)	151.1 (20)	340.6 (16)	296.7 (16)	396.9 (17)	290.8 (17)	075.3 (18)	273.9 (29)
min degree/max prev. exp. dist.	315.5 (36)	292.9 (35)	104.7 (34)	510.2 (38)	018.3 (34)	101.5 (35)	094.6 (37)	219.6 (38)	442.6 (35)	385.2 (34)	475.8 (38)	340.8 (34)	102.0 (35)	410.2 (43)
min degree/max total bound gap	329.1 (41)	306.3 (41)	111.9 (40)	523.0 (39)	018.8 (41)	107.3 (43)	102.0 (48)	249.4 (48)	464.4 (41)	401.2 (41)	504.6 (45)	362.2 (43)	104.5 (39)	317.9 (38)
min degree/median exp. dist.	305.1 (32)	300.4 (36)	100.4 (30)	479.9 (25)	018.3 (35)	094.2 (30)	082.5 (22)	196.3 (32)	428.2 (30)	369.5 (28)	437.5 (27)	313.5 (22)	098.5 (29)	245.3 (22)
min degree/min exp. dist.	488.9 (53)	720.8 (53)	244.0 (53)	739.8 (53)	030.2 (53)	219.4 (53)	158.8 (53)	330.8 (53)	643.1 (53)	551.3 (53)	806.5 (53)	565.6 (53)	152.7 (53)	1638.4 (53)
min degree/random index	299.3 (31)	275.1 (27)	098.9 (27)	487.4 (28)	017.7 (24)	093.7 (29)	082.7 (23)	190.2 (30)	429.8 (32)	376.0 (33)	439.4 (28)	321.9 (30)	098.5 (28)	244.6 (21)
max prev. exp. dist./max degree	122.0 (02)	124.1 (05)	043.8 (02)	197.1 (03)	009.2 (04)	043.5 (03)	044.0 (04)	080.6 (02)	177.7 (03)	142.6 (02)	186.4 (06)	119.3 (02)	047.1 (03)	**118.2** (01)
max prev. exp. dist./min degree	316.3 (37)	291.2 (34)	104.9 (35)	504.6 (34)	018.0 (31)	101.1 (34)	093.3 (36)	216.7 (37)	443.9 (36)	389.4 (36)	472.2 (37)	344.3 (38)	101.9 (34)	443.8 (46)
max prev. exp. dist./max exp. dist.	249.7 (19)	222.6 (20)	084.4 (19)	420.5 (19)	014.2 (18)	077.8 (18)	067.8 (17)	147.3 (19)	359.9 (20)	313.7 (20)	403.9 (20)	298.2 (20)	075.9 (19)	249.7 (24)
max prev. exp. dist./max prev. exp. dist.	293.4 (25)	268.7 (25)	115.4 (47)	480.9 (26)	019.4 (46)	102.4 (36)	089.2 (32)	183.7 (24)	416.1 (23)	363.5 (24)	461.7 (32)	343.3 (37)	097.5 (25)	336.1 (42)
max prev. exp. dist./max total bound gap	332.3 (43)	313.2 (43)	112.5 (42)	534.5 (46)	019.5 (47)	108.2 (46)	096.9 (43)	229.6 (45)	479.6 (48)	412.9 (48)	506.7 (47)	366.8 (46)	106.3 (44)	438.2 (45)
max prev. exp. dist./median exp. dist.	321.8 (38)	329.3 (46)	107.1 (36)	507.2 (35)	018.5 (37)	105.9 (40)	094.7 (38)	224.7 (43)	445.1 (37)	389.6 (37)	467.2 (36)	339.3 (33)	105.7 (42)	458.5 (47)
max prev. exp. dist./min exp. dist.	487.6 (51)	718.4 (51)	241.6 (52)	738.0 (52)	030.1 (51)	217.6 (51)	158.0 (51)	328.9 (50)	642.0 (52)	550.6 (52)	802.5 (50)	563.0 (50)	152.0 (52)	1613.6 (51)
max prev. exp. dist./random index	312.4 (34)	289.9 (33)	100.8 (32)	507.9 (36)	018.1 (32)	099.2 (33)	089.0 (31)	208.6 (33)	446.0 (38)	388.8 (35)	466.7 (34)	342.2 (36)	102.7 (37)	304.2 (35)
max total bound gap/max degree	**103.3** (01)	**093.7** (01)	**039.1** (01)	**159.1** (01)	**007.7** (01)	**037.8** (01)	**038.4** (01)	**074.3** (01)	**140.3** (01)	**137.9** (01)	**143.8** (01)	**109.8** (01)	**044.7** (01)	139.0 (09)
max total bound gap/min degree	334.2 (47)	305.4 (39)	112.4 (41)	525.7 (40)	018.7 (38)	107.9 (44)	100.3 (47)	249.3 (47)	458.0 (39)	401.0 (40)	500.4 (43)	361.5 (42)	105.4 (41)	320.6 (40)
max total bound gap/max exp. dist.	267.3 (21)	235.7 (21)	091.4 (21)	441.1 (21)	014.8 (20)	084.0 (21)	072.1 (20)	159.4 (21)	372.4 (21)	327.5 (21)	429.6 (25)	316.9 (25)	078.6 (21)	284.4 (32)
max total bound gap/max prev. exp. dist.	332.4 (44)	310.5 (42)	115.3 (46)	530.0 (44)	019.1 (44)	108.0 (45)	098.3 (44)	226.9 (44)	475.0 (46)	408.4 (44)	508.2 (48)	366.2 (45)	107.8 (48)	419.0 (44)
max total bound gap/max total bound gap	337.1 (48)	313.8 (44)	112.8 (43)	538.8 (48)	019.4 (45)	106.6 (41)	096.3 (42)	221.6 (40)	471.2 (45)	410.9 (46)	504.8 (46)	370.4 (48)	106.8 (45)	318.1 (39)
max total bound gap/median exp. dist.	333.6 (46)	349.9 (47)	114.7 (45)	529.0 (43)	018.9 (42)	110.6 (47)	099.8 (46)	238.6 (46)	465.7 (42)	409.7 (45)	491.4 (40)	357.5 (39)	106.9 (46)	314.4 (37)
max total bound gap/min exp. dist.	487.1 (50)	717.1 (50)	240.6 (50)	737.8 (51)	030.0 (50)	217.4 (50)	157.2 (49)	329.0 (51)	641.9 (50)	550.0 (50)	804.0 (52)	564.0 (52)	151.7 (50)	1602.1 (50)
max total bound gap/random index	327.6 (39)	303.0 (38)	108.8 (37)	530.5 (45)	018.4 (36)	105.1 (39)	095.0 (39)	221.0 (39)	462.7 (40)	400.4 (39)	489.6 (39)	359.3 (40)	104.8 (40)	295.6 (34)
random index/max degree	232.9 (14)	216.2 (16)	079.3 (14)	334.2 (13)	013.0 (13)	073.4 (13)	065.0 (15)	136.2 (13)	303.6 (13)	240.1 (13)	268.7 (14)	202.5 (14)	067.6 (13)	260.6 (28)
random index/min degree	297.7 (29)	275.2 (28)	098.7 (25)	487.8 (29)	017.9 (30)	093.3 (27)	084.8 (30)	187.2 (26)	429.2 (31)	369.2 (27)	432.9 (26)	320.8 (28)	098.2 (27)	244.2 (20)
random index/max exp. dist.	236.1 (15)	206.7 (14)	081.1 (15)	400.8 (15)	013.7 (15)	075.2 (15)	065.0 (14)	141.3 (16)	332.3 (15)	294.1 (15)	381.1 (15)	285.0 (15)	072.8 (15)	239.3 (19)
random index/max prev. exp. dist.	307.6 (33)	285.1 (31)	102.9 (33)	509.7 (37)	018.1 (33)	097.5 (32)	089.4 (33)	210.0 (35)	441.8 (34)	390.8 (38)	466.8 (35)	341.0 (35)	101.4 (33)	305.0 (36)
random index/max total bound gap	328.1 (40)	306.2 (40)	110.0 (39)	528.3 (42)	018.7 (40)	104.1 (38)	095.1 (40)	222.0 (41)	466.1 (43)	405.8 (43)	495.2 (41)	360.6 (41)	106.0 (43)	293.4 (33)
random index/median exp. dist.	296.4 (26)	288.9 (32)	100.5 (31)	478.0 (24)	017.8 (26)	091.8 (24)	083.0 (26)	192.8 (31)	421.4 (26)	359.5 (23)	426.6 (24)	313.5 (23)	098.1 (26)	258.8 (27)
random index/min exp. dist.	487.8 (52)	718.5 (52)	241.3 (51)	737.8 (50)	030.1 (52)	217.6 (52)	158.0 (50)	329.3 (52)	641.9 (51)	550.5 (51)	802.8 (51)	563.1 (51)	151.7 (51)	1614.7 (52)
random index/random index	292.5 (24)	264.5 (24)	095.8 (23)	476.2 (23)	017.8 (27)	091.4 (23)	081.4 (21)	183.8 (25)	421.0 (25)	367.8 (26)	425.6 (23)	314.0 (24)	096.9 (23)	224.6 (14)
max bound gap	248.6 (18)	218.0 (18)	084.2 (17)	417.3 (18)	014.1 (17)	078.2 (19)	066.5 (16)	145.5 (17)	354.8 (19)	307.6 (18)	398.5 (18)	298.1 (19)	075.0 (16)	248.9 (23)
max total degree	159.8 (08)	147.1 (07)	062.7 (11)	243.5 (09)	010.3 (07)	049.6 (08)	054.9 (12)	098.6 (08)	220.1 (11)	192.5 (10)	191.4 (07)	121.7 (04)	057.6 (10)	160.9 (12)
min total degree	297.3 (28)	275.2 (29)	099.3 (28)	491.3 (33)	017.8 (28)	092.9 (26)	084.1 (28)	188.1 (27)	426.9 (28)	369.7 (29)	440.9 (30)	321.9 (31)	099.2 (31)	250.8 (25)
max combined total bound gap	332.7 (45)	316.6 (45)	113.8 (44)	536.3 (47)	018.9 (43)	107.0 (42)	098.6 (45)	224.3 (42)	476.2 (47)	412.0 (47)	503.6 (44)	367.2 (47)	106.9 (47)	321.5 (41)
random pair	291.2 (23)	264.2 (23)	096.2 (24)	481.6 (27)	017.4 (22)	092.1 (25)	083.6 (27)	182.9 (23)	417.5 (24)	365.1 (25)	423.8 (22)	317.3 (26)	096.9 (24)	223.5 (13)

Coloring by column:

Worst strategy — Random pair — Best strategy

Next, we will discuss the most important observations backed by evidence from Tables 3.1 to 3.3.

Observation 1. There are better strategies than simply choosing a *random pair*.

Evidence: The best rank *random pair* achieves is 13^{th} in Table 3.3 on the dataset *Unbalance*. Often it ranks around the mid-twenties in Tables 3.1 to 3.3. This means that there are (many) strategies that perform better than *random pair*.

Observation 2. *Max total bound gap / max degree* is the best strategy for earlier rounds.

Evidence: The ranked scores of strategy *max total bound gap / max degree* are highlighted in Table 3.4. For rounds $2 \cdot M_{MST}$ up to $7 \cdot M_{MST}$, the strategy ranks the best out of all evaluated strategies. When one has really limited labeling capabilities, this strategy performs very well across all datasets. It always has the best AUC score out of all tested strategies, except for the dataset *Unbalance* (see Table 3.3).

Table 3.4: Highlighted ranks: The ranks of the strategy *max total bound gap / max degree* from Tables 3.1 and 3.2.

	1 M_{MST}	2 M_{MST}	3 M_{MST}	4 M_{MST}	5 M_{MST}	6 M_{MST}	7 M_{MST}	8 M_{MST}	9 M_{MST}	10 M_{MST}
Average performance	06	01	01	01	01	01	01	08	08	08
Borda count	09	01	01	01	01	01	02	08	08	08

Observation 3. *Max degree* is generally a good criterion, especially in the earlier rounds.

Evidence: In Tables 3.1 to 3.3 a lot of green cells belong to a strategy with *max degree*. This means that it performs close to or equal to the best performance. Thus, it is a good strategy to choose at least one of the indices based on *max degree*. Especially in the earlier rounds. In round $3 \cdot M_{MST}$, strategies with *max degree* rank in Table 3.1: (10^{th}, 7^{th}, 4^{th}, 6^{th}, 2^{nd}, 5^{th}, 13^{th}, 9^{th}, 12^{th}, 11^{th}, 3^{rd}, 1^{st}, 14^{th}). Thus, the entire top 14 is filled by strategies with *max degree* except for the eight place, which is obtained by *max total degree*. This criterion is thus highly effective in the earlier rounds.

Observation 4. *Min exp. distance* and *median exp. distance* are bad criteria.

Evidence: Both *min exp. distance* and *median exp. distance* perform terrible. After $10 \cdot M_{MST}$, strategies with *min exp. distance* and with *median exp. distance* are ranked (23^{rd}, 49^{th}, 53^{rd}, 51^{st}, 50^{th}, 52^{nd}) and (16^{th},

48^{th}, 41^{st}, 47^{th}, 46^{th}, 42^{nd}), respectively in Table 3.1. Only combining with *max degree* can save the performance. *Min exp. distance* is for all other combinations colored red in Tables 3.1 to 3.3, which means that it is (or close to) the worst performance.

Observation 5. Although the prediction is directly dependent on the bound gap, *max bound gap* is only a good strategy after $> 7 \cdot M_{MST}$ rounds.

Evidence: The ranked scores of strategy *max bound gap* are highlighted in Table 3.5. In the early rounds (up to $4 \cdot M_{MST}$), this strategy performs even worse than *random pair*. After that, it quickly becomes one of the best performing strategies, even ranking first in the later rounds. Due to the slow start, the AUC scores are remarkably mediocre, see Table 3.3.

Table 3.5: Highlighted ranks: The ranks of the strategy *max bound gap* from Tables 3.1 and 3.2.

	1 M_{MST}	2 M_{MST}	3 M_{MST}	4 M_{MST}	5 M_{MST}	6 M_{MST}	7 M_{MST}	8 M_{MST}	9 M_{MST}	10 M_{MST}
Average performance	25	26	22	19	18	17	10	02	01	01
Borda count	23	23	22	23	17	15	05	01	01	01

Observation 6. *Max exp. distance* is a late bloomer.

Evidence: Whilst *min exp. distance* and *median exp. distance* perform bad, *max exp. distance* gets increasingly better. Comparing the ranks in Table 3.1 in round $5 \cdot M_{MST}$ with round $10 \cdot M_{MST}$ gives:

$$\begin{array}{cccccc} (4^{\text{th}} & 14^{\text{th}} & 15^{\text{th}} & 18^{\text{th}} & 21^{\text{st}} & 13^{\text{th}}) \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ (11^{\text{th}} & 3^{\text{rd}} & 7^{\text{th}} & 2^{\text{nd}} & 5^{\text{th}} & 4^{\text{th}}) \end{array}$$

Only *max degree / max exp. distance* loses terrain. After round $10 \cdot M_{MST}$, the top 7 contains five strategies with *max exp. distance*, which is noteworthy.

Observation 7. Performance is relatively robust across datasets (AUC scores).

Evidence: In Table 3.3, every strategy has approximately the same color across datasets. This means that the relative performance is not very dependent on the dataset. However, *Unbalance* gives the most deviant results. This implies that the balancedness of the dataset could influence the performance of a strategy.

3.8 Real world experiment

In order to test if the observations also hold for *real world* datasets, we also evaluate the strategies on the *cifar10* [92] and *mnist* [101] datasets. These datasets consist of images of ten different categories. To limit memory space and running time, we only take the first 1,000 samples of the training set for each dataset. The distance between two images is determined by the Euclidean norm, which was also used in the previous experiments. The results can be found in Table 3.6, where the *average performance* is given (see Section 3.6.4).

Next, we discuss (using Table 3.6) if the observations from Section 3.7 also hold for these real world datasets. Still, there are many better strategies than simply choosing a *random pair* (Observation 1). *Max total bound gap / max degree* also remains the best strategy for earlier rounds (Observation 2), but now the performance falls off after $2 \cdot M_{MST}$ rounds. *Max degree* is generally a good criterion (Observation 3). The best strategies often use this criterion. *Min exp. distance* and *median exp. distance* are still bad criteria (Observation 4). But now, *max bound gap* is not a good strategy even after $> 7 \cdot M_{MST}$ rounds (Observation 5). After $10 \cdot M_{MST}$ rounds, it ranks 30^{th}, whilst simply selecting a random pair ranks 20^{th}. Perhaps, this strategy needs even more rounds to become good. *Max exp. distance* is also not longer a late bloomer (Observation 6), as multiple strategies with this criterion rank higher after $10 \cdot M_{MST}$ rounds, then after $5 \cdot M_{MST}$ rounds. Furthermore, it ranks worse after $10 \cdot M_{MST}$ rounds compared with the previous experiment. Perhaps, this strategy also needs more rounds to start blooming. We believe that the difference could be explained by the dimensionality of the datasets. The *cifar10* and *mnist* dataset have a higher dimensionality ($32 \times 32 \times 3$) and (28×28), respectively. It is well-known that in higher dimensional space, most points will be far away. Therefore, dimensionality could play a role in the distribution of pairwise distances. This in turn, could have an effect on some strategies such as *max bound gap* and *max exp. distance*, which is why we believe that these strategies may need more time to start performing well on these datasets. The AUC performance remains relatively stable for these datasets (Observation 7).

In general, most previous observations still hold for these real world datasets. Only some strategies that previously performed well in the later rounds, did not start improving as well on these datasets. It could be that more rounds are necessary.

Table 3.6: Average performance (cifar10 & mnist): For each dataset (*cifar10* & *mnist*) and repetition, the prediction error of a strategy is averaged in rounds $i \cdot M_{MST}$ with $i \in \{1, \ldots, 10\}$. The ranking (by column) of each average prediction error is noted in brackets. Coloring of each column is done linearly between the worst and baseline (random pair) score and linearly between the baseline (random pair) and the best score.

Strategy	1 M_{MST}	2 M_{MST}	3 M_{MST}	4 M_{MST}	5 M_{MST}	6 M_{MST}	7 M_{MST}	8 M_{MST}	9 M_{MST}	10 M_{MST}
max degree/max degree	12.961 (08)	7.957 (05)	6.469 (04)	5.568 (06)	4.905 (03)	**4.358** (01)	4.098 (03)	3.891 (04)	3.581 (02)	3.389 (03)
max degree/min degree	12.529 (06)	7.847 (04)	**6.328** (01)	5.554 (05)	**4.889** (01)	4.567 (04)	4.273 (08)	4.043 (09)	3.823 (09)	3.721 (09)
max degree/max exp. dist.	13.199 (10)	9.023 (10)	6.625 (07)	6.075 (11)	5.400 (11)	5.093 (11)	4.230 (06)	4.030 (08)	3.645 (05)	3.542 (06)
max degree/max prev. exp. dist.	11.332 (04)	8.231 (09)	6.871 (10)	5.695 (08)	4.992 (05)	4.655 (07)	4.449 (10)	4.234 (11)	4.055 (11)	3.910 (12)
max degree/max total bound gap	14.655 (11)	10.614 (13)	6.375 (02)	5.656 (07)	5.105 (07)	4.927 (10)	4.662 (11)	3.930 (05)	3.772 (07)	3.667 (07)
max degree/median exp. dist.	11.391 (05)	7.789 (03)	6.440 (03)	5.425 (04)	4.893 (02)	4.370 (03)	**4.033** (01)	3.807 (02)	**3.560** (01)	3.413 (04)
max degree/min exp. dist.	12.574 (07)	10.236 (12)	8.898 (13)	7.819 (14)	6.944 (14)	6.603 (14)	5.918 (14)	5.638 (14)	4.881 (13)	4.189 (13)
max degree/random index	13.170 (09)	9.060 (11)	7.501 (12)	5.373 (02)	5.090 (06)	4.620 (05)	4.260 (07)	3.947 (07)	3.794 (08)	3.688 (08)
linked/max degree	17.355 (14)	**7.641** (01)	7.075 (11)	5.376 (03)	5.114 (08)	4.367 (02)	4.186 (05)	**3.722** (01)	3.628 (03)	3.371 (02)
linked/min degree	20.476 (45)	20.332 (36)	20.074 (34)	19.679 (32)	19.132 (31)	18.451 (28)	17.655 (25)	16.776 (24)	15.839 (24)	14.900 (24)
linked/max exp. dist.	20.436 (17)	20.246 (17)	19.938 (17)	19.520 (17)	18.980 (19)	18.361 (19)	17.676 (28)	16.975 (31)	16.268 (31)	15.571 (31)
linked/max prev. exp. dist.	20.434 (15)	20.230 (16)	19.929 (16)	19.501 (16)	18.992 (20)	18.431 (24)	17.822 (36)	17.181 (36)	16.501 (36)	15.867 (36)
linked/max total bound gap	20.476 (47)	20.323 (29)	20.131 (38)	19.870 (39)	19.524 (40)	19.100 (42)	18.614 (43)	18.078 (43)	17.495 (44)	16.859 (42)
linked/median exp. dist.	20.440 (18)	20.249 (18)	19.942 (18)	19.520 (18)	18.973 (18)	18.368 (21)	17.664 (27)	16.892 (27)	15.931 (26)	14.787 (19)
linked/min exp. dist.	20.459 (29)	20.392 (49)	20.333 (51)	20.284 (51)	20.224 (51)	20.164 (52)	20.113 (53)	20.056 (53)	19.992 (53)	19.929 (53)
linked/random index	20.435 (16)	20.228 (15)	19.896 (15)	19.441 (15)	18.841 (15)	18.129 (15)	17.269 (15)	16.408 (15)	15.505 (15)	14.655 (15)
min degree/max degree	11.146 (03)	8.016 (06)	6.715 (08)	**5.297** (01)	4.978 (04)	4.714 (09)	4.067 (02)	3.873 (03)	3.655 (06)	**3.367** (01)
min degree/min degree	20.475 (44)	20.333 (38)	20.079 (37)	19.688 (35)	19.143 (33)	18.457 (29)	17.646 (23)	16.751 (22)	15.819 (22)	14.883 (22)
min degree/max exp. dist.	20.469 (31)	20.324 (30)	20.064 (29)	19.669 (29)	19.126 (30)	18.441 (27)	17.635 (21)	16.785 (25)	15.910 (25)	15.044 (26)
min degree/max prev. exp. dist.	20.470 (34)	20.328 (35)	20.074 (35)	19.698 (37)	19.187 (37)	18.535 (36)	17.785 (32)	16.970 (30)	16.133 (29)	15.323 (29)
min degree/max total bound gap	20.476 (46)	20.347 (39)	20.152 (44)	19.895 (46)	19.571 (46)	19.181 (47)	18.734 (48)	18.235 (48)	17.694 (48)	17.125 (48)
min degree/median exp. dist.	20.469 (33)	20.326 (33)	20.073 (32)	19.687 (34)	19.144 (34)	18.465 (31)	17.655 (26)	16.745 (21)	15.729 (18)	14.671 (16)
min degree/min exp. dist.	20.482 (53)	20.426 (53)	20.366 (53)	20.306 (53)	20.239 (53)	20.174 (53)	20.101 (52)	20.031 (52)	19.957 (52)	19.890 (52)
min degree/random index	20.471 (35)	20.328 (34)	20.071 (31)	19.678 (31)	19.135 (32)	18.435 (25)	17.639 (22)	16.744 (20)	15.804 (21)	14.889 (23)
max prev. exp. dist./max degree	11.034 (02)	8.138 (08)	6.758 (09)	6.105 (12)	5.294 (10)	4.657 (08)	4.172 (04)	3.934 (06)	3.633 (04)	3.481 (05)
max prev. exp. dist./min degree	20.468 (30)	20.326 (32)	20.074 (33)	19.691 (36)	19.171 (36)	18.525 (35)	17.758 (30)	16.957 (28)	16.117 (28)	15.318 (28)
max prev. exp. dist./max exp. dist.	20.457 (27)	20.291 (27)	20.009 (26)	19.593 (25)	19.068 (25)	18.458 (30)	17.791 (34)	17.111 (34)	16.443 (35)	15.798 (35)
max prev. exp. dist./max prev. exp. dist.	20.456 (25)	20.285 (24)	20.000 (24)	19.600 (26)	19.114 (28)	18.536 (37)	17.913 (37)	17.273 (37)	16.630 (37)	15.988 (37)
max prev. exp. dist./max total bound gap	20.474 (36)	20.350 (43)	20.151 (43)	19.879 (42)	19.535 (43)	19.108 (43)	18.609 (41)	18.066 (41)	17.486 (42)	16.863 (43)
max prev. exp. dist./median exp. dist.	20.453 (19)	20.277 (19)	19.989 (20)	19.577 (23)	19.057 (24)	18.440 (26)	17.786 (33)	17.078 (33)	16.364 (33)	15.599 (32)
max prev. exp. dist./min exp. dist.	20.477 (50)	20.407 (50)	20.328 (49)	20.246 (49)	20.151 (49)	20.049 (49)	19.943 (49)	19.832 (50)	19.705 (50)	19.577 (50)
max prev. exp. dist./random index	20.459 (28)	20.298 (28)	20.022 (28)	19.624 (28)	19.106 (27)	18.468 (32)	17.768 (31)	17.059 (32)	16.323 (32)	15.624 (33)
max total bound gap/max degree	**10.005** (01)	7.721 (02)	6.537 (06)	6.005 (10)	5.709 (12)	5.496 (13)	5.312 (13)	5.175 (13)	5.066 (14)	4.977 (14)
max total bound gap/min degree	20.476 (49)	20.347 (41)	20.154 (45)	19.891 (45)	19.559 (45)	19.164 (45)	18.709 (47)	18.201 (47)	17.656 (47)	17.070 (47)
max total bound gap/max exp. dist.	20.474 (39)	20.351 (45)	20.157 (46)	19.879 (43)	19.525 (41)	19.095 (40)	18.606 (40)	18.065 (40)	17.459 (40)	16.816 (40)
max total bound gap/max prev. exp. dist.	20.474 (38)	20.347 (40)	20.149 (41)	19.879 (44)	19.540 (44)	19.117 (44)	18.618 (44)	18.084 (44)	17.493 (43)	16.875 (44)
max total bound gap/max total bound gap	20.474 (37)	20.357 (48)	20.170 (47)	19.912 (48)	19.578 (47)	19.182 (48)	18.706 (46)	18.177 (46)	17.591 (46)	16.967 (46)
max total bound gap/median exp. dist.	20.476 (48)	20.351 (44)	20.148 (39)	19.869 (38)	19.515 (39)	19.085 (38)	18.587 (38)	18.030 (38)	17.432 (38)	16.783 (38)
max total bound gap/min exp. dist.	20.479 (52)	20.422 (52)	20.362 (52)	20.297 (52)	20.227 (52)	20.158 (51)	20.081 (51)	20.008 (51)	19.931 (51)	19.852 (51)
max total bound gap/random index	20.475 (41)	20.347 (42)	20.148 (40)	19.871 (40)	19.512 (38)	19.087 (39)	18.594 (39)	18.037 (39)	17.448 (39)	16.804 (39)
random index/max degree	16.248 (13)	11.815 (14)	8.985 (14)	7.204 (13)	6.047 (13)	5.294 (12)	4.745 (12)	4.374 (12)	4.089 (12)	3.876 (11)
random index/min degree	20.469 (32)	20.324 (31)	20.068 (30)	19.675 (30)	19.122 (29)	18.425 (23)	17.607 (19)	16.722 (19)	15.792 (20)	14.879 (21)
random index/max exp. dist.	20.456 (26)	20.284 (23)	20.002 (25)	19.585 (24)	19.035 (23)	18.361 (20)	17.611 (20)	16.812 (26)	16.031 (27)	15.269 (27)
random index/max prev. exp. dist.	20.455 (22)	20.290 (26)	20.014 (27)	19.607 (27)	19.088 (26)	18.480 (34)	17.815 (35)	17.127 (35)	16.418 (34)	15.721 (34)
random index/max total bound gap	20.475 (42)	20.352 (46)	20.151 (42)	19.875 (41)	19.528 (42)	19.098 (41)	18.610 (42)	18.066 (42)	17.467 (41)	16.825 (41)
random index/median exp. dist.	20.455 (20)	20.281 (20)	19.984 (19)	19.556 (20)	18.967 (17)	18.284 (17)	17.510 (18)	16.675 (18)	15.709 (17)	14.689 (17)
random index/min exp. dist.	20.477 (51)	20.410 (51)	20.332 (50)	20.250 (50)	20.157 (50)	20.061 (50)	19.948 (50)	19.829 (49)	19.700 (49)	19.570 (49)
random index/random index	20.456 (23)	20.287 (25)	19.997 (23)	19.552 (19)	18.965 (16)	18.260 (16)	17.443 (16)	16.555 (16)	15.619 (16)	14.699 (18)
max bound gap	20.456 (24)	20.283 (21)	19.997 (22)	19.574 (22)	19.029 (22)	18.369 (22)	17.680 (29)	16.962 (29)	16.228 (30)	15.524 (30)
max total degree	14.995 (12)	8.042 (07)	6.510 (05)	5.781 (09)	5.247 (09)	4.631 (06)	4.363 (09)	4.140 (10)	3.903 (10)	3.742 (10)
min total degree	20.475 (43)	20.332 (37)	20.075 (36)	19.686 (33)	19.149 (35)	18.473 (33)	17.654 (24)	16.759 (23)	15.834 (23)	14.903 (25)
max combined total bound gap	20.474 (40)	20.356 (47)	20.171 (48)	19.909 (47)	19.579 (48)	19.176 (46)	18.701 (45)	18.166 (45)	17.581 (45)	16.946 (45)
random pair	20.455 (21)	20.284 (22)	19.994 (21)	19.566 (21)	18.993 (21)	18.298 (18)	17.495 (17)	16.644 (17)	15.733 (19)	14.819 (20)

Coloring by column:

3.8.1 Performance max degree

An important observation from both Section 3.7 and Table 3.6, is that *max degree* is a good criterion. The best performing methods often include this criterion. We briefly want to discuss why we believe that choosing a sample that has been already chosen often (max degree) is beneficial. In order to predict the actual distance, a lower and upper bound is established using the triangle inequality (Section 3.4.2). When the distance is labeled between i and j, the triangle inequality can be used to derive information about the distances between i and k if the distance between j and k is known. Therefore, labeling a sample with the highest degree, gives a lot of possible triangle inequality combinations that can be made, which could provide much information. This is why we believe that this criterion performs really well.

3.9 Discussion and future research

This research can be viewed as a pioneering contribution and is a significant first step in APDL. Below we elaborate on both the shortcomings of the approach proposed, and the related challenges for further research.

Perfect expert: It is assumed that the expert does not make any mistake in determining the distance between two instances. This is a common, yet unreasonably optimistic, assumption in AL research. Settles [156] states that "we have often assumed that there is a single infallible annotator whose labels can be trusted" and views this assumption as one of the six practical challenges for AL. How to deal with a noisy expert remains a critical research problem. A way of mitigating the mistakes of the expert in APDL is to allow some ϵ-boundary around the labels and incorporating this into the approximation bounds. Still, there are many more ways to deal with an imperfect expert, which should be investigated. Using properties of a metric, mistakes can be spotted and reevaluated.

Underlying distance metric: In all experiments, the Euclidean distance was used as underlying distance metric. This might affect the conclusions that were drawn, as alternative distance metrics might be favorable for different strategies. In future research, this could be investigated by changing the underlying distance metric and evaluating if the same strategies are always performing the best.

Complex strategies: In our research, we have examined many selection algorithms based on straightforward criteria. Newer and more complex

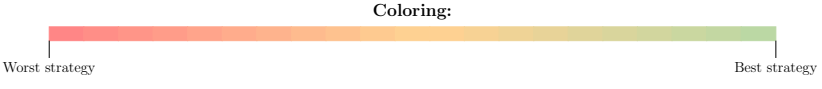
strategies could be developed, reducing the prediction error even more. Consider for example mixing strategies, where one strategy works well in the beginning (e.g., *max total bound gap / max degree*) and switch to another strategy (e.g., *max bound gap*) that works better later on. Another way, would be to select each round a specific strategy with a certain probability. Additionally, *transfer learning* [62, 201] can be applied to train an even more advanced model (e.g., a neural network) using labeled datasets. Such a model can be trained to choose a good strategy at a specific time, where the new prediction error can be used to either reward or penalize the selection. If the chosen strategy selected a pair that gave a lot of insight, the model can be updated to select this strategy more often in similar cases. When properly trained, the model could be applied to new datasets to determine the selection strategy. Whether this is a good approach, depends on the ability of the model to transfer the learned information over to the new dataset.

Running time: In this research, we have used straightforward criteria that are easy to compute. However, when more complex strategies are designed, *running time* could start to play a role. The importance of running time is mostly task dependent. The cost of coming up with the next query should be balanced with the cost of the labeling done by the expert. We consider APDL to be particularly useful in situations where the expert can only be queried a limited number of times (due to high costs). However, running time is something that should be considered in future work when more complex strategies are used. When a strategy is too hard to compute, approximation algorithms could be developed. The average running time of each strategy can be seen in Table 3.7. We believe that the difference in running time can mostly be explained by the following phenomenon. When there are more samples that satisfy the selection criterion, a random selection is made between these samples. This function takes more time, when there are more samples to choose from. Consider, for example, the difference between *random index / max degree* and *random index / min degree* that take on average 685 and 978 seconds, respectively. There are considerably more samples with the same *minimum* degree compared to the *maximum* degree. In Table 3.7, we observe that strategies consistently are slower when they have more samples that satisfy the criterion.

Space complexity: In the experiments, at most $M = 1{,}000$ samples were used, as this already leads to 499,500 different pairs. To store the approximation bounds for each pair, $\mathcal{O}(M^2)$ is necessary. This can quickly become infeasible for large M. Although rather time expensive, these

Table 3.7: Average running time: The total running time of each strategy averaged over all repetitions and datasets (including cifar10 & mnist). The ranking is noted in brackets. Coloring is done linearly between the worst and best score.

Strategy	Time (s)	Strategy	Time (s)	Strategy	Time (s)
max degree/max degree	0720 (16)	max degree/min degree	0729 (19)	max degree/max exp. dist.	0678 (09)
max degree/max prev. exp. dist.	0741 (20)	max degree/max total bound gap	0701 (13)	max degree/median exp. dist.	0681 (11)
max degree/min exp. dist.	0706 (14)	max degree/random index	0678 (10)	linked/max degree	0722 (18)
linked/min degree	1009 (41)	linked/max exp. dist.	1247 (49)	linked/max prev. exp. dist.	1170 (47)
linked/max total bound gap	0947 (36)	linked/median exp. dist.	0915 (32)	linked/min exp. dist.	0583 (05)
linked/random index	0995 (40)	min degree/max degree	0751 (21)	min degree/min degree	1052 (45)
min degree/max exp. dist.	1228 (48)	min degree/max prev. exp. dist.	0889 (27)	min degree/max total bound gap	0891 (29)
min degree/median exp. dist.	1070 (46)	min degree/min exp. dist.	0583 (04)	min degree/random index	1022 (43)
max prev. exp. dist./max degree	0710 (15)	max prev. exp. dist./min degree	0902 (30)	max prev. exp. dist./max exp. dist.	1303 (51)
max prev. exp. dist./max prev. exp. dist.	1352 (53)	max prev. exp. dist./max total bound gap	0868 (24)	max prev. exp. dist./median exp. dist.	0828 (22)
max prev. exp. dist./min exp. dist.	0606 (07)	max prev. exp. dist./random index	0944 (35)	max total bound gap/max degree	0661 (08)
max total bound gap/min degree	0921 (34)	max total bound gap/max exp. dist.	1274 (50)	max total bound gap/max prev. exp. dist.	0882 (26)
max total bound gap/max total bound gap	0917 (33)	max total bound gap/median exp. dist.	0837 (23)	max total bound gap/min exp. dist.	0575 (02)
max total bound gap/random index	0889 (28)	random index/max degree	0685 (12)	random index/min degree	0978 (39)
random index/max exp. dist.	1321 (52)	random index/max prev. exp. dist.	0878 (25)	random index/max total bound gap	0911 (31)
random index/median exp. dist.	0960 (38)	random index/min exp. dist.	0577 (03)	random index/random index	0956 (37)
max bound gap	1017 (42)	max total degree	0720 (17)	min total degree	1044 (44)
max combined total bound gap	0598 (06)	random pair	**0563** (01)		



approximation bounds could be calculated every time they are needed. Yet, for large problems, a better solution is necessary. A major insight of this research is that choosing based on *max degree* consistently performs well. This criterion does not use any information from the approximation bounds, which is why this is ideal for large problems, as the approximation bounds are only necessary for the final predictions. More research is necessary to optimize large APDL problems.

Using feature values: It was assumed in Section 3.2.2 that no feature values should be used. In this way, the observations from this research are not dependent on the application domain. Furthermore, if new methods are developed that do use feature values, our tested selection strategies can function as a good baseline. Adding information (using the feature values) should only increase the performance of an APDL method. Thus, when a model is performing worse than any one of our suggested strategies, it should be considered as a major warning sign. Additionally, during the APDL process, a model could be used to evaluate if the feature values could help the prediction. If so, feature values could be introduced into the query selection after some rounds.

Gaining insight: Demystifying AL can give us critical insights. Which samples are useful to query? Can we understand why? Can we explain why certain selection algorithms perform better? Is the clusteredness/balancedness of a dataset relevant? Are there better indicators for the usefulness of a sample query? Answering these kinds of questions could lead to better performing models.

Error reduction rate: The reduction rate in prediction error instigates many exciting research opportunities. Can guarantees be derived about the speed with which the prediction error converges for certain strategies? It would be especially useful for practical applications to know how many labels should be gathered to get at most a prediction error of $\delta > 0$. To derive such a guarantee, either theoretical proof or substantial numerical evidence is necessary. Additionally, the effect of a tight or loose initial upper bound for the maximum distance on the convergence speed could also be investigated.

Additional application: We think that APDL can also be used to determine the complexity of a dataset. When a strategy needs more rounds to attain a certain prediction error, the dataset might be more complex, as it is harder to learn the pairwise distances. In this way, APDL can even be useful for fully labeled datasets. Which strategies to use and how complexity is exactly quantified with APDL are all interesting subjects for future research.

Prediction model: Recall that there are two critical components in APDL, namely 'Which pair is queried each round?' and 'How to use this information to make the best prediction?' The focus of our research was to answer the first question. To make a prediction of a distance, we used the upper and lower bound approximation and took the average as prediction (see Definition 3.6.1). Therein lies a large opportunity for improvement, as a more advanced prediction model could improve the final prediction as well as the query selection. Using a tuned weighted average of the upper and lower approximation could already perform better.

3.10 Summary

We started by introducing the problem of APDL, where the goal is to actively learn the pairwise distances between all instances. We established upper and lower bound approximations using properties of a distance function. Furthermore, we presented an update rule that automatically updates the upper and lower bounds using the newest labeled distance. Then, we provided

fourteen selection criteria, which gave us 53 query strategies combined. These strategies do not use feature values, making the observations from the experiments domain-independent. This makes these selection strategies ideal candidates for a baseline in future research.

The experiments led to valuable new insights. These observations were tested by evaluating all strategies on two real world datasets (*cifar10* & *mnist*). We found multiple strategies that perform better than simply randomly selecting a pair (Observation 1). This shows that it is indeed possible to 'smartly' select the indices. We determined that the performance of the strategies was not very dependent on the datasets (Observation 7). The performance only changed somewhat in a highly unbalanced case. We identified *max degree* to be a consistently good criterion. In Section 3.8.1, we explained why we believe that this criterion is useful. Consequently, we also discovered which strategies should not be chosen due to general bad performance (Observation 4). Choosing the right selection strategy could potentially save many hours and resources. The findings from the experiments are not dependent on the dimensionality of the data or (noisy) feature values, as feature values were not taken into account. However, more dimensions could lead to higher sparsity (curse of dimensionality), which is why a mix of sparse and dense datasets were used.

Part II

Benchmarking Binary Prediction Models

Chapter 4

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

Contents

Based on *Etienne van de Bijl, Jan Klein, Joris Pries, Sandjai Bhulai, Mark Hoogendoorn, and Rob van der Mei* (2022): "**The Dutch Draw: Constructing a universal baseline for binary prediction models**". Under revision. [14]

Abstract

Novel prediction methods should always be compared to a baseline to know how well they perform. Without this frame of reference, the performance score of a model is basically meaningless. What does it mean when a model achieves an F_1 of 0.8 on a test set? A proper baseline is needed to evaluate the 'goodness' of a performance score. Comparing with the latest state-of-the-art model is usually insightful. However, being state-of-the-art can change rapidly when newer models are developed. Contrary to an advanced model, a simple dummy classifier could be used. However, the latter 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. We present a universal baseline method for all *binary classification* models, named the *Dutch Draw* (DD). This approach weighs simple classifiers and determines the best classifier to use as a baseline. We theoretically derive the DD 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 DD baseline is: (1) *general*, as it is applicable to any binary classification problems; (2) *simple*, as it is quickly determined without training or parameter-tuning; (3) *informative*, as insightful conclusions can be drawn from the results. The DD baseline serves two purposes. First, to enable comparisons across research papers by this robust and universal baseline. Secondly, to provide a sanity check during the development process of a prediction model. It is a major warning sign when a model does not outperform the DD baseline.

4.1 Introduction

A typical data science project can be crudely simplified to 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 [189]. Before deploying a new model, it should be tested whether it meets certain predefined success criteria. A baseline plays an essential role in this evaluation, as it gives an indication of the actual performance of a model.

However, which baseline should be selected? A good baseline is desirable, but what explicitly makes a baseline 'good'? Comparing with the latest state-of-the-art model is usually insightful. However, being state-of-the-art can change rapidly when newer models are developed. Reproducibility of 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 it hard or even impossible to compare older results with newer research. 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 should have: *generality*, *simplicity*, and *informativeness*.

Generality In research, a new model is commonly compared to a limited number of existing models that are 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 either zero or one. Here, one could already select a decision tree [119], random forest [40], variants of naive Bayes [182], k-nearest neighbors [1], support vector machine [157], neural network [170], or logistic regression model [154] to evaluate the performance. These models are often trained specifically for a problem instance with parameters tuned for optimal performance in that specific case. Hence, these methods are not general. One could not take a decision tree that is used for determining bankruptcy [119] and use it as a baseline for a pathological voice detection problem [122]. 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. But, it is hard to determine for a measure if a baseline is too complex or not. Essentially, two components are critical in our view: (1) *computational time* and (2) *explainability*. It is necessary for practical applications that the baseline can 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 it expected that a new model is outperformed by this ingeniously complicated baseline, or is it exactly what one would expect? This leads to the last property of a good standard baseline.

Informativeness A baseline should be informative. 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 any 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 obviously beaten by any developed model. If not, this should be considered a major warning sign.

Our research focuses on finding such a general, simple and informative baseline for *binary classification* problems. However, 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 it is not always informative. The information gained by performing better than a simple dummy classifier

can even 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. [91] determined for a large family of binary performance measures that the optimal classifier consists of a sign function with a threshold tailored to each specific measure. To determine the optimal classifier, it is necessary to know or approximate $\mathbb{P}(Y = 1|X = x)$, which is the probability that the binary label Y is 1 given the features $X = x$. Lipton et al. [106] had 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 classifier and the optimal threshold classifier have their strengths and weaknesses. In this chapter, we introduce a novel baseline approach, named the *Dutch Draw* (DD). The DD eliminates these weaknesses, whilst keeping their strengths. The DD can be seen as a dummy classifier on steroids. Instead of arbitrarily choosing a dummy classifier, we mathematically derive which classifier, from a family of classifiers, has the best expected performance. Also, this expected performance can be directly determined, making it really 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; (6) and 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 how the DD baseline can be used in practice to identify when models should definitely be reconsidered; (4) and we made the DD available in a Python package [134].

4.2 Preliminaries

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

4.2.1 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 $M \in \mathbb{N}_{>0}$ observations, let $\mathcal{M} := \{1, \ldots, M\}$ be the set of observation indices. Each instance, denoted by $\mathbf{x}_i$, has $K \in \mathbb{N}_{>0}$ explanatory feature values. These features can be categorical or numerical. Without loss of generality, we assume that $\mathbf{x}_i \in \mathbb{R}^K$ for all $i \in \mathcal{M}$. Moreover, each observation has a corresponding output value $y_i \in \{0, 1\}$. Now, let $\mathbf{X} := [\mathbf{x}_1 \ldots \mathbf{x}_M]^T \in \mathbb{R}^{M \times K}$ denote the matrix with all observations and their explanatory feature values and let $\mathbf{y} = (y_1, \ldots, y_M) \in \{0, 1\}^M$ 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.

4.2.2 Evaluation measures

An *evaluation measure* quantifies the prediction performance of a trained model. We categorize the evaluation measures into two groups: *base measures* and *performance measures* [28]. 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 measures are a function of one or more these four base measures. To shorten notation, let $\hat{P} := \mathrm{TP} + \mathrm{FP}$ and $\hat{N} := \mathrm{TN} + \mathrm{FN}$ denote the number of positively and negatively predicted instances respectively.

All considered performance measures and base measures are shown in Table 4.1. Also their abbreviations, possibly alternative names, their definitions and corresponding codomains are presented in Table 4.1. 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 4.3, the case-specific

codomains are provided when we discuss the evaluation measures in more detail. Finally, note that the list is not exhaustive, but it contains most of the commonly used evaluation measures.

Table 4.1: Definitions and codomains of evaluation measures

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

Ill-defined measures

Not every evaluation measure is well-defined. Often, the problem occurs due to division by zero. For example, the *True Positive Rate* (TPR) defined as $\mathrm{TPR} = \mathrm{TP}/P$ cannot be calculated whenever $P = 0$. Therefore, we have made assumptions for the allowed values of P, N, $\hat{P}$, and $\hat{N}$. These are shown in Table 4.2. One exception is the *Prevalence Threshold* (PT)

[11], where the denominator is zero if TPR is equal to the False Positive Rate (defined as FPR = FP/N). Depending on the classifier, this situation could occur regularly. Therefore, PT is omitted throughout the rest of this research.

Table 4.2: Assumptions on domains P, N, $\hat{P}$ and $\hat{N}$: Some measures are not defined if P, N, $\hat{P}$ or $\hat{N}$ is equal to zero. These domain requirements are therefore necessary (always $M > 0$).

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

4.3 Dutch Draw

In this section, we introduce the Dutch Draw (DD) framework and discuss how this method is able to 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 4.1. First, we provide the family of DD classifiers, and thereafter we explain how the optimal classifier generates the baseline.

4.3.1 Dutch Draw classifiers

The goal of our research is to provide a universal baseline for any evaluation measure in binary classification. The DD baseline comes from choosing the best DD classifier. Before we discuss what 'best' actually entails, we have to define the DD classifier in general. This is the function $\sigma_\theta : \mathbb{R}^{M \times K} \to \{0, 1\}^M$ with input an evaluation dataset with M observations and K feature values per observation. The function generates the predictions for these observations by outputting a vector of M binary predictions. It is described in words

as:

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

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

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

with $(\mathbf{1}_E(i))_{i \in \mathcal{M}}$ the vector with ones in the positions in E and zeroes elsewhere. Note that 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 $\hat{N}_\theta := M - \lfloor M \cdot \theta \rceil$. To be specific, these two numbers are integers, and thus, different values of θ can lead to the same value of $\hat{P}_\theta$. Therefore, we introduce the parameter $\theta^* := \frac{\lfloor M \cdot \theta \rceil}{M}$ as the 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 4.1 (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 4.2.2) 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:

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

where $\mathcal{D}(\mathrm{TP}_\theta)$ is the domain of TP_θ. The definition of this domain is given in Equation (4.1).

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

$$\mathrm{TP}_\theta \sim \mathrm{Hypergeometric}(M, P, \lfloor M \cdot \theta \rceil),$$

$$\mathrm{FP}_\theta \sim \mathrm{Hypergeometric}(M, N, \lfloor M \cdot \theta \rceil),$$

$$\mathrm{FN}_\theta \sim \mathrm{Hypergeometric}(M, P, M - \lfloor M \cdot \theta \rceil),$$

$$\mathrm{TN}_\theta \sim \mathrm{Hypergeometric}(M, N, M - \lfloor M \cdot \theta \rceil).$$

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 before, $\mathrm{TP}_\theta + \mathrm{FN}_\theta = P$ and $\mathrm{TN}_\theta + \mathrm{FP}_\theta = N = M - P$, and we also have $\mathrm{TP}_\theta + \mathrm{FP}_\theta = \lfloor M \cdot \theta \rceil$, 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:

$$\mathrm{TP}_\theta = \mathrm{TP}_\theta, \tag{B1}$$

$$\mathrm{FP}_\theta = \hat{P}_\theta - \mathrm{TP}_\theta, \tag{B2}$$

$$\mathrm{FN}_\theta = P - \mathrm{TP}_\theta, \tag{B3}$$

$$\mathrm{TN}_\theta = N - \hat{P}_\theta + \mathrm{TP}_\theta, \tag{B4}$$

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

Example: distribution F_β score To illustrate how the probability function $f_{X_\theta}(a, b)$ can directly be derived, we consider the F_β *score* $\mathrm{F}_\theta^{(\beta)}$ [39]. It is the *weighted harmonic average* between the *True Positive Rate* (TPR_θ) and the *Positive Predictive Value* (PPV_θ). The latter two performance measures are discussed extensively in Sections 4.A.5 and 4.A.9, respectively. The F_β score balances predicting the actual positive observations correctly (TPR_θ) and being cautious in predicting observations as positive (PPV_θ). The factor $\beta > 0$ indicates how much more TPR_θ is weighted compared to PPV_θ. The F_β score is commonly defined as:

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

By substituting PPV_θ and TPR_θ by their definitions (see Table 4.1) and using Equations (B1) and (B2), 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 depends 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 0, 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 $\mathcal{D}(\mathrm{TP}_\theta)$ be the domain of TP_θ, then:

$$\mathcal{D}(\mathrm{TP}_\theta) := \big\{ i \in \mathbb{N}_0 : \max\{0, \lfloor M \cdot \theta \rceil - (M-P)\} \leq i \leq \min\{P, \lfloor M \cdot \theta \rceil\} \big\}. \quad (4.1)$$

Consequently, the range of $X_\theta\,(a,b)$ is given by

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

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

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

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

$$\mathbb{E}[X_\theta\,(a,b)] = a \cdot \mathbb{E}[\mathrm{TP}_\theta] + b = a \cdot \frac{\lfloor M \cdot \theta \rceil}{M} \cdot P + b. \quad (\square)$$

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 measure, we again consider $\mathrm{F}_\theta^{(\beta)}$.

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:

$$\mathbb{E}[\mathrm{F}_\theta^{(\beta)}] = \mathbb{E}\left[X_\theta\left(\frac{1+\beta^2}{\beta^2 \cdot P + \lfloor M \cdot \theta \rceil}, 0\right)\right] \overset{(\square)}{=} \frac{1+\beta^2}{\beta^2 \cdot P + \lfloor M \cdot \theta \rceil} \cdot \mathbb{E}[\mathrm{TP}_\theta] + 0$$

$$= \frac{\lfloor M \cdot \theta \rceil \cdot P \cdot (1+\beta^2)}{M \cdot (\beta^2 \cdot P + \lfloor M \cdot \theta \rceil)} = \frac{(1+\beta^2) \cdot P \cdot \theta^*}{\beta^2 \cdot P + M \cdot \theta^*}. \quad (4.2)$$

A full overview of the distribution and mean of all considered base and performance measures is given in Table 4.3. All the calculations performed to derive the corresponding distributions and expectations are provided in Appendix A.

4.3.2 Optimal Dutch Draw classifier

Next, we discuss how the DD baseline will ultimately be derived. In order to do so, an overview is presented in Figure 4.1. Starting with the definition of the DD classifiers in Section 4.3.1 and determining their expectations for commonly used measures (see Table 4.3), we are now able to identify the *optimal* DD classifier. Given a performance measure and dataset, the optimal DD classifier is found by optimizing (taking the minimum or maximum of) the associated expectation for $\theta \in [0, 1]$.

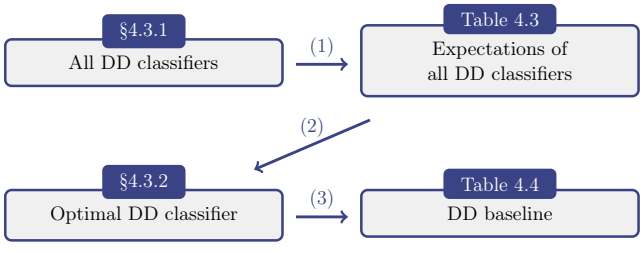


Figure 4.1: Road to DD baseline: This is an overview of how the DD baseline is determined. (1) all expectations are derived; (2) the expectation is maximized/minimized; (3) the performance of the best DD classifier is the DD baseline.

Dutch Draw baseline

The optimal DD classifiers and the corresponding DD baseline can be found in Table 4.4. For many performance measures, it is optimal to always predict positive or 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 measures, almost all parameter values give the optimal baseline. Next, we give an example to illustrate how the results of Table 4.4 are derived.

Example: DD baseline for the F_β score To determine the DD baseline, the extreme values of the expectation $\mathbb{E}[\mathrm{F}_\theta^{(\beta)}]$ need to be identified. To do

Table 4.3: Properties of performance measures for a DD classifier: Expectation and distribution of each performance measure for a DD classifier σ_θ with $\theta^* = \frac{\lfloor M \cdot \theta \rfloor}{M}$.

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

this, examine the following function $f : [0, 1] \to [0, 1]$ defined as:

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

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

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

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

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

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

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

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

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

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

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

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

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

Non-linear performance measures We have shown that the DD baseline is straightforward for performance measures that can be written in *linear* terms of TP_θ. However, there are measures, such as the $\mathrm{G}_\theta^{(2)}$, where this is not possible. This could make it hard to derive a closed-form expression for the maximum expectation. Previously, we have seen in Table 4.4 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 show the optimal $\theta^*_{\max}$ in Figure 4.2 for a fixed M and increasing P. This shows that $\theta = 0.5$ is not always the optimal value. The optimal value is significantly different when $P \ll N$ or $P \gg N$. We believe that this can be explained by the following reasoning: Observe that $\mathrm{G}^{(2)} = \sqrt{\mathrm{TPR} \cdot \mathrm{TNR}}$ is zero when

Table 4.4: DD baseline: For many evaluation measures, the minimum and maximum expected score of all allowed DD classifiers is determined, which is the DD baseline. In this table, the baselines and the optimizing parameters are given. "-" denotes that no closed-form expression was found.

Measure	$\max\{\mathbb{E}\}$	$\Theta^\star_{\max} := \arg\max\{\mathbb{E}\}$	$\min\{\mathbb{E}\}$	$\Theta^\star_{\min} := \arg\min\{\mathbb{E}\}$
TP	P	$\{1\}$	0	$\{0\}$
TN	$M-P$	$\{0\}$	0	$\{1\}$
FN	P	$\{0\}$	0	$\{1\}$
FP	$M-P$	$\{1\}$	0	$\{0\}$
TPR	1	$\{1\}$	0	$\{0\}$
TNR	1	$\{0\}$	0	$\{1\}$
FNR	1	$\{0\}$	0	$\{1\}$
FPR	1	$\{1\}$	0	$\{0\}$
PPV	$\frac{P}{M}$	$\Theta^* \setminus \{0\}$	$\frac{P}{M}$	$\Theta^* \setminus \{0\}$
NPV	$1-\frac{P}{M}$	$\Theta^* \setminus \{1\}$	$1-\frac{P}{M}$	$\Theta^* \setminus \{1\}$
FDR	$1-\frac{P}{M}$	$\Theta^* \setminus \{0\}$	$1-\frac{P}{M}$	$\Theta^* \setminus \{0\}$
FOR	$\frac{P}{M}$	$\Theta^* \setminus \{1\}$	$\frac{P}{M}$	$\Theta^* \setminus \{1\}$
F_β	$\frac{(1+\beta^2)\cdot P}{\beta^2\cdot P+M}$	$\{1\}$	$\frac{(1+\beta^2)\cdot P}{M(\beta^2\cdot P+1)}$	$\{\frac{1}{M}\}$
J	0	Θ^*	0	Θ^*
MK	0	$\Theta^* \setminus \{0,1\}$	0	$\Theta^* \setminus \{0,1\}$
Acc	$\max\{\frac{P}{M}, 1-\frac{P}{M}\}$	$\{[P < \frac{M}{2}]\}$ [2]	$\min\{\frac{P}{M}, 1-\frac{P}{M}\}$	$\{[P > \frac{M}{2}]\}$ [2]
BAcc	$\frac{1}{2}$	Θ^*	$\frac{1}{2}$	Θ^*
MCC	0	$\Theta^* \setminus \{0,1\}$	0	$\Theta^* \setminus \{0,1\}$
κ	0	Θ^* [3]	0	Θ^* [3]
FM	$\sqrt{\frac{P}{M}}$	$\{1\}$	$\frac{\sqrt{P}}{M}$	$\{\frac{1}{M}\}$
$G^{(2)}$	-	-	0	$\{0,1\}$
TS	$\frac{P}{M}$	$\{1\}$	0	$\{0\}$

[2] If $P = \frac{M}{2}$, then Θ^*. Note that *Iverson brackets* are used to simplify notation.

[3] If $P = M$, then $\Theta^* \setminus \{1\}$.

either TPR or TNR is zero, which is the minimum score. When there are few positive labels, it must therefore 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 Dutch Draw baseline can still be derived for non-linear performance measures by determining the expectations of *all* DD classifiers. However, this can be greatly improved in future research (see Section 4.5).

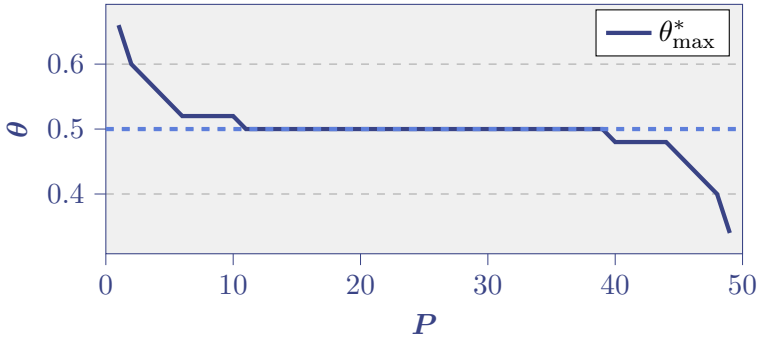


Figure 4.2: Non-trivial $\theta^*_{\max}$ for $G^{(2)}_\theta$ For each $P \in \{1, \ldots, 49\}$, the optimal $\theta^*_{\max}$ is derived for the performance measure $G^{(2)}_\theta$ with a dataset consisting of P positive and $50 - P$ negative samples. This shows that the optimal value is not straightforward.

4.4 Dutch Draw in practice

Now that we have established how to derive the DD baseline, it is time to see how the Dutch Draw could be used in practice. A data scientist (DS) is given a dataset (*Cleveland heart disease*) to predict whether patients have a heart disease given several feature values. The DS randomly splits the dataset into a training (90%) and test set (10%) and chooses the F_1 measure to evaluate how well a model performs. The Dutch Draw baseline (of the test set) immediately provides a performance reference (0.735) for any model that the DS is going to use. The DS trains two common machine learning algorithms (*decision tree* and *k-nearest neighbors*) with default parameters in *scikit-learn* [128]. To get a good estimation of the expected performance, an average is taken over 10 iterations. They achieve an average score of 0.727 and 0.710, respectively, which are worse than the Dutch Draw baseline (0.735). This is a major warning sign. Although the decision tree performed better than k-nearest neighbors, it should still not be used. Thus, the DS decides to train three other models (*logistic regression*, *random forest*, and

Gaussian naive Bayes), which end up performing better than the baseline. The DS decides to use the logistic regression model in practice, as this model achieves the highest score *and* beats the Dutch Draw baseline. An overview of the performance of these five models and the baseline is shown in Table 4.5 (including other performance measures).

4.4.1 Example: Cleveland Heart Disease

The objective of this dataset is to predict whether patients have a heart disease given several feature values. In order to do so, we used five commonly used machine learning algorithms to perform this binary classification task: *logistic regression*, *decision tree*, *random forest*, *k-nearest neighbors*, and *Gaussian naive Bayes*. These algorithms all had their default parameters in *scikit-learn* [128]. The dataset was randomly split in a training (90%) and test set (10%). Table 4.5 shows the corresponding performance results.

Before applying a newly developed model to actual patients, its performance should at least be better than the DD baseline, as the latter does not learn anything from the feature values of the data. In Table 4.5, we see that some methods fail to beat the baseline and should therefore be reconsidered. For example, *decision tree* and *k-nearest neighbors* underperform for the F_β *score* (FBETA), *Fowlkes-Mallows Index* (FM), and *Threat Score* (TS). Note that the two methods were not trained to be optimal for the selected performance measures, whereas the DD does take the performance measure into account. However, this does not make the comparison unfair, since they are not competing for being the best prediction method. After all, the DD baseline is a minimal requirement for any new binary classification method. Even though a model is optimized for, say, the Accuracy, its performance should still beat the DD baseline for the F_1 score, as both the Accuracy and F_1 score provide indications of the overall prediction performance. To conclude, this example shows how the DD can be used in practice and why it is valuable in the evaluation process.

4.5 Discussion and conclusion

In this research, we have proposed a new baseline methodology named the *Dutch Draw* (DD). 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; (6) and an explainable minimal requirement for any new model. We have shown that for most commonly used measures the DD baseline can be theoretically determined

Table 4.5: Comparing performance to the DD baseline: The average result (10 iterations) of five standard machine learning algorithms (decision tree (DT), k-nearest neighbors (KNN), *logistic regression (LR)*, *random forest (RF)*, and *Gaussian naive Bayes (GNB)*) on the *Cleveland Heart Disease* dataset for many commonly used performance measures. The first two columns are the minimal and maximal DD baseline (DDB). The cases, where the baseline is not beaten, are in bold. Note that all performance measures (except FDR and FOR) are maximized in these comparisons.

			Model				
Measure	DDB-min	DDB-max	DT	KNN	LR	RF	GNB
PPV	0.581	0.581	0.800	0.846	0.941	1.000	0.882
NPV	0.419	0.419	0.625	0.611	0.857	0.867	0.786
FDR	0.419	0.419	0.200	0.154	0.059	0.000	0.118
FOR	0.581	0.581	0.375	0.389	0.143	0.133	0.214
ACC	0.419	0.581	0.710	0.710	0.903	0.935	0.839
BACC	0.500	0.500	0.718	0.729	0.906	0.944	0.840
FBETA	0.061	0.735	**0.727**	**0.710**	0.914	0.941	0.857
MCC	0.000	0.000	0.430	0.457	0.805	0.878	0.674
J	0.000	0.000	0.436	0.457	0.812	0.889	0.679
MK	0.000	0.000	0.425	0.457	0.798	0.867	0.668
KAPPA	0.000	0.000	0.422	0.434	0.803	0.870	0.672
FM	0.137	0.762	**0.730**	**0.719**	0.915	0.943	0.857
G2	0.000	0.500	0.716	0.719	0.906	0.943	0.840
TS	0.000	0.581	**0.571**	**0.550**	0.842	0.889	0.750

Coloring:

Minimum range — Maximum range

(see Table 4.4). When the baseline cannot be derived directly, it can be identified quickly by computation. For most performance measures, the DD baseline reduces to one of the following three cases: (I) always predicting positive or negative; (II) always predicting positive or negative, except for one instance; (III) any DD classifier, except maybe for $\theta^* = 0$ or $\theta^* = 1$. However, there are exceptions to these three cases, as we have seen in Figure 4.2. This showed that the DD does not always reduce 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 minimal requirement for any novel model to at least beat the DD baseline. When this does not happen, the question is raised how much a new method has even learned from the data, 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 specific domain to obtain additional insights. In Section 4.4, we have shown how the DD should be used in practice and that commonly used approaches such as *k-nearest neighbors* and a *decision tree* can underperform. Hence, using the Dutch Draw as a general, simple and informative baseline should be the new gold standard in any binary model evaluation process.

4.5.1 Further research

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

Firstly, we are now able to determine whether a binary classification model performs better than a 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 easily 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 quite 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 performance of these models be quantified? Is it only a slightly better model or is it a leap forward?

Secondly, 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, such that comparing a score to the baseline is not necessary anymore. There are many ways how the DD baseline can be used to scale a measure. Let $\Delta_{\max}$ and $\Delta_{\min}$ denote the maximum and minimum Dutch Draw 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 Dutch Draw baseline ($\Delta_{\min}$) gets value -1, because every Dutch Draw classifier is then performing 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 Dutch Draw baseline. All scores above $\Delta_{\max}$ are scaled to $[0, 1]$. In this case, the performance at least performed better than the best Dutch Draw baseline.

Thirdly, 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.

Fourthly, for some (less straightforward) performance measures, 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 therefore be investigated.

As a final note, we have published the code for the DD, such that the reader can easily implement the baseline into their binary classification problems [134].

Appendix (Chapter 4)

4.A Mathematical derivations

This section contains the complete theoretical analysis that is used to gather the information presented in Sections 4.2 and 4.3, and more specifically, Tables 4.2 to 4.4. Each subsection is dedicated to one of the evaluation measures. 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 4.A.1.

Table 4.A.1: Overview of the Appendix: Each measure is discussed in the corresponding section in the Appendix.

Measure	TP	TN	FN	FP	TPR	TNR	FNR	FPR	PPV	NPV	FDR	FOR
Section	4.A.1	4.A.2	4.A.3	4.A.4	4.A.5	4.A.6	4.A.7	4.A.8	4.A.9	4.A.10	4.A.11	4.A.12
Measure	F_β	J	MK	Acc	BAcc	MCC	κ	FM	$G^{(2)}$	PT	TS	
Section	4.A.13	4.A.14	4.A.15	4.A.16	4.A.17	4.A.18	4.A.19	4.A.20	4.A.21	4.A.22	4.A.23	

4.A.1 Number of True Positives

The *number of True Positives* TP_θ is one of the four base measures that are introduced in Section 4.2.2. 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.

Definition and distribution

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

$$\mathrm{TP}_\theta \overset{(B1)}{=} X_\theta\left(1,0\right) \sim f_{X_\theta}\left(1,0\right).$$

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

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

Expectation

The expectation of TP_θ using the DD is given by

$$\mathbb{E}[\mathrm{TP}_\theta] = \mathbb{E}[X_\theta\left(1,0\right)] \overset{(\square)}{=} \frac{\lfloor M \cdot \theta \rceil}{M} \cdot P = \theta^* \cdot P. \tag{4.3}$$

Optimal baselines

The DD baseline is given by the optimal expectation. Equation (4.3) shows that the expected value depends on the parameter θ. Therefore, either the minimum or maximum of the expectation yields the baseline. They are given by

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

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

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

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

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

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

$$\theta^*_{\min} \in \arg\min_{\theta^* \in \Theta^*} \{\mathbb{E}[\mathrm{TP}_{\theta^*}]\} = \arg\min_{\theta^* \in \Theta^*} \{\theta^*\} = \{0\},$$

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

4.A.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 4.2.2. This base measure counts the number of negative predicted instances that are actually negative.

Definition and distribution

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

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

which corresponds to Equation (B4). Furthermore,

$$\mathrm{TN}_\theta \overset{(B4)}{=} X_\theta \left(1, M - P - \lfloor M \cdot \theta \rceil\right) \sim f_{X_\theta} \left(1, M - P - \lfloor M \cdot \theta \rceil\right),$$

and for its range

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

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

$$\mathbb{E}[\mathrm{TN}_\theta] = \mathbb{E}\left[X_\theta \left(1, M - P - \lfloor M \cdot \theta \rceil\right)\right] \overset{(\square)}{=} 1 \cdot \mathbb{E}[\mathrm{TP}_\theta] + M - P - \lfloor M \cdot \theta \rceil$$

$$= \left(1 - \frac{\lfloor M \cdot \theta \rceil}{M}\right)(M - P) = (1 - \theta^*)(M - P).$$

Optimal baselines

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

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

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

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

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

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

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

$$\theta^*_{\min}\in\arg\min_{\theta^*\in\Theta^*}\{\mathbb{E}[\mathrm{TN}_{\theta^*}]\} = \arg\min_{\theta^*\in\Theta^*}\{1-\theta^*\} = \{1\},$$

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

4.A.3 Number of False Negatives

The *number of False Negatives* FN_θ is one of the four base measures that are introduced in Section 4.2.2. This base measure counts the number of mistakes made by predicting instances negative while the actual labels are positive.

Definition and distribution

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

$$\mathrm{FN}_\theta \overset{(B3)}{=} P - \mathrm{TP}_\theta = X_\theta(-1,P) \sim f_{X_\theta}(-1,P),$$

and for its range:

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

Expectation

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

$$\mathbb{E}[\mathrm{FN}_\theta] = \mathbb{E}[X_\theta\,(-1, P)] \overset{(\square)}{=} -1 \cdot \mathbb{E}[\mathrm{TP}_\theta] + P = \left(1 - \frac{\lfloor M \cdot \theta \rceil}{M}\right) \cdot P$$

$$= (1 - \theta^*) \cdot P.$$

Optimal baselines

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

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

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

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

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

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

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

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

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

4.A.4 Number of False Positives

The *number of False Positives* FP_θ is one of the four base measures that we discussed in Section 4.2.2. This base measure counts the number of mistakes made by predicting instances positive while the actual labels are negative.

Definition and distribution

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

$$\mathrm{FP}_\theta \overset{(B2)}{=} \lfloor M \cdot \theta \rceil - \mathrm{TP}_\theta = X_\theta\left(-1, \lfloor M \cdot \theta \rceil\right) \sim f_{X_\theta}\left(-1, \lfloor M \cdot \theta \rceil\right),$$

and for its range:

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

Expectation

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

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

Optimal baselines

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

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

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

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

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

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

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

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

$$\theta^*_{\max} \in \arg\max_{\theta^* \in \Theta^*} \{\mathbb{E}[\mathrm{FP}_{\theta^*}]\} = \arg\max_{\theta^* \in \Theta^*} \{\theta^*\} = \{1\}.$$

4.A.5 True Positive Rate

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

Definition and distribution

The True Positive Rate is commonly defined as

$$\mathrm{TPR}_\theta = \frac{\mathrm{TP}_\theta}{P}. \tag{4.4}$$

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

$$\mathrm{TPR}_\theta = X_\theta\left(\frac{1}{P}, 0\right) \sim f_{X_\theta}\left(\frac{1}{P}, 0\right), \tag{4.5}$$

and for its range:

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

Expectation

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

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

Optimal baselines

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

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

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

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

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

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

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

$$\theta^*_{\min} \in \arg\min_{\theta^*\in\Theta^*} \{\mathbb{E}[\mathrm{TPR}_{\theta^*}]\} = \arg\min_{\theta^*\in\Theta^*} \{\theta^*\} = \{0\},$$

$$\theta^*_{\max} \in \arg\max_{\theta^*\in\Theta^*} \{\mathbb{E}[\mathrm{TPR}_{\theta^*}]\} = \arg\max_{\theta^*\in\Theta^*} \{\theta^*\} = \{1\},$$

respectively.

4.A.6 True Negative Rate

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

Definition and distribution

The True Negative Rate is commonly defined as

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

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

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

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

$$\mathrm{TNR}_\theta = X_\theta \left(\frac{1}{M-P}, 1 - \frac{\lfloor M \cdot \theta \rceil}{M-P} \right) \sim f_{X_\theta} \left(\frac{1}{M-P}, 1 - \frac{\lfloor M \cdot \theta \rceil}{M-P} \right), \tag{4.6}$$

and for its range:

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

4

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

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

Optimal baselines

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

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

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

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

$$\theta_{\min} \in \arg\min_{\theta\in[0,1]} \{\mathbb{E}[\mathrm{TNR}_\theta]\} = \arg\min_{\theta\in[0,1]} \left\{1 - \frac{\lfloor M\cdot\theta\rfloor}{M}\right\} = \left[1-\frac{1}{2M}, 1\right],$$

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

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

$$\theta^*_{\min} \in \arg\min_{\theta^*\in\Theta^*} \{\mathbb{E}[\mathrm{TNR}_{\theta^*}]\} = \arg\min_{\theta^*\in\Theta^*} \{1-\theta^*\} = \{1\},$$

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

4.A.7 False Negative Rate

The *False Negative Rate* FNR_θ or *Miss Rate* is the performance measure that indicates the relative number of incorrectly predicted positive observations. Therefore, it can be seen as the counterpart to the True Positive Rate that is discussed in Section 4.A.5.

Definition and distribution

The False Negative Rate is commonly defined as

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

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

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

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

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

and for its range:

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

Expectation

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

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

Optimal baselines

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

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

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

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

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

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

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

$$\theta^*_{\min} \in \arg\min_{\theta^* \in \Theta^*} \{\mathbb{E}[\mathrm{FNR}_{\theta^*}]\} = \arg\min_{\theta^* \in \Theta^*} \{1 - \theta^*\} = \{1\},$$

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

4.A.8 False Positive Rate

The *False Positive Rate* FPR_θ or *Fall-out* is the performance measure that shows the fraction of incorrectly predicted negative observations. Hence, it can be seen as the counterpart to the True Negative Rate that is introduced in Section 4.A.6.

Definition and distribution

The False Positive Rate is commonly defined as

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

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

$$\mathrm{FPR}_\theta = \frac{\lfloor M \cdot \theta \rceil - \mathrm{TP}_\theta}{M - P}. \tag{4.7}$$

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

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

with range:

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

Expectation

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

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

Optimal baselines

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

$$\min_{\theta \in [0,1]} \{\mathbb{E}[\mathrm{FPR}_\theta]\} = \min_{\theta \in [0,1]} \left\{\frac{\lfloor M \cdot \theta \rceil}{M}\right\} = 0,$$

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

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

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

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

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

$$\theta^*_{\min} \in \arg\min_{\theta^*\in\Theta^*} \{\mathbb{E}[\mathrm{FNR}_{\theta^*}]\} = \arg\min_{\theta^*\in\Theta^*} \{\theta^*\} = \{0\},$$

$$\theta^*_{\max} \in \arg\max_{\theta^*\in\Theta^*} \{\mathbb{E}[\mathrm{FNR}_{\theta^*}]\} = \arg\max_{\theta^*\in\Theta^*} \{\theta^*\} = \{1\}.$$

4.A.9 Positive Predictive Value

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

Definition and distribution

The Positive Predictive Value is commonly defined as

$$\mathrm{PPV}_\theta = \frac{\mathrm{TP}_\theta}{\mathrm{TP}_\theta + \mathrm{FP}_\theta}. \tag{4.8}$$

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

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

Note that this performance measure is only defined whenever $\lfloor M\cdot\theta \rceil > 0$, otherwise the denominator is zero. Therefore, we assume specifically for

PPV_θ that $\theta \geq \frac{1}{2M}$. The definition of PPV_θ is linear in TP_θ and can thus be formulated as

$$\mathrm{PPV}_\theta = X_\theta\left(\frac{1}{\lfloor M \cdot \theta \rceil}, 0\right) \sim f_{X_\theta}\left(\frac{1}{\lfloor M \cdot \theta \rceil}, 0\right), \tag{4.9}$$

with range:

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

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

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

Optimal baselines

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

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

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

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

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

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

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

4.A.10 Negative Predictive Value

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

Definition and distribution

The *Negative Predictive Value* is commonly defined as

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

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

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

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

$$\begin{aligned}\mathrm{NPV}_\theta = X_\theta \left(\frac{1}{M - \lfloor M \cdot \theta \rceil}, 1 - \frac{P}{M - \lfloor M \cdot \theta \rceil} \right) \\ \sim f_{X_\theta} \left(\frac{1}{M - \lfloor M \cdot \theta \rceil}, 1 - \frac{P}{M - \lfloor M \cdot \theta \rceil} \right),\end{aligned} \tag{4.10}$$

with range:

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

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

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

Optimal baselines

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

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

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

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

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

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

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

4.A.11 False Discovery Rate

The *False Discovery Rate* FDR_θ is the performance measure that looks at the fraction of positively predicted observations that are actually negative. Therefore, it can be seen as the counterpart to the Positive Predictive Value that we discuss in Section 4.A.9. Consequently, a small value means the model is cautious in predicting observations as positive, while a large value means the opposite.

Definition and distribution

The *False Discovery Rate* is commonly defined as

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

With the help of Equation (4.9), this definition can be rewritten as

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

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

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

with range:

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

Expectation

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

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

Optimal baselines

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

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

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

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

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

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

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

4.A.12 False Omission Rate

The *False Omission Rate* FOR_θ is the performance measure that considers the fraction of observations that are predicted negative, but are in fact positive. Hence, it can be seen as the counterpart to the Negative Predictive Value that is introduced in Section 4.A.10. As a consequence, a small value means the model is cautious is predicting observations negatively, while a large value means the opposite.

Definition and distribution

The *False Omission Rate* is commonly defined as

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

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

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

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

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

with range:

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

Expectation

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

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

Optimal baselines

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

$$\min_{\theta \in [0, 1-1/(2M))} \{\mathbb{E}[\mathrm{FOR}_\theta]\} = \frac{P}{M},$$

$$\max_{\theta \in [0, 1-1/(2M))} \{\mathbb{E}[\mathrm{FOR}_\theta]\} = \frac{P}{M},$$

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

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

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

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

4.A.13 F_β Score

The *F_β score* $\mathrm{F}_\theta^{(\beta)}$ was introduced by Chinchor [39]. It is the weighted harmonic average between the True Positive Rate (TPR_θ) and the Positive Predictive Value (PPV_θ). These two performance measures are discussed extensively in Sections 4.A.5 and 4.A.9, respectively, and their summarized results are shown in Tables 4.3 and 4.4. The F_β score balances predicting the actual positive observations correctly (TPR_θ) and being cautious in predicting observations as positive (PPV_θ). The factor $\beta > 0$ indicates how much more TPR_θ is weighted compared to PPV_θ.

Definition and distribution

The F_β score is commonly defined as

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

By using the definitions of TPR_θ and PPV_θ in Equations (4.4) and (4.8), $\mathrm{F}_\theta^{(\beta)}$ can be formulated in terms of the base measures:

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

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

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

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

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

with range:

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

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

$$\mathbb{E}[\mathrm{F}_\theta^{(\beta)}] = \mathbb{E}\left[X_\theta \left(\frac{1+\beta^2}{\beta^2 \cdot P + \lfloor M \cdot \theta \rceil}, 0 \right) \right] \overset{(\square)}{=} \frac{1+\beta^2}{\beta^2 \cdot P + \lfloor M \cdot \theta \rceil} \cdot \mathbb{E}[\mathrm{TP}_\theta] + 0$$

$$= \frac{\lfloor M \cdot \theta \rceil \cdot P \cdot (1+\beta^2)}{M \cdot (\beta^2 \cdot P + \lfloor M \cdot \theta \rceil)} = \frac{(1+\beta^2)\cdot P \cdot \theta^*}{\beta^2 \cdot P + M \cdot \theta^*}. \tag{4.11}$$

Optimal baselines

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

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

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

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

It is strictly positive for all t in its domain, thus f is strictly increasing in t. This means $\mathbb{E}[\mathrm{F}_\theta^{(\beta)}]$ given in Equation (4.11) is non-decreasing in both θ and θ^*. This is because the term $\lfloor M\cdot\theta\rceil/M$ is non-decreasing in θ. Hence, the extreme values of the expectation of $\mathrm{F}_\theta^{(\beta)}$ are its border values:

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

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

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

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

$$= \begin{cases} [\frac{1}{2},1] & \text{if } M=1 \\ [\frac{1}{2M},\frac{3}{2M}) & \text{if } M>1, \end{cases}$$

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

$$= \begin{cases} [\frac{1}{2},1] & \text{if } M=1 \\ [1-\frac{1}{2M},1] & \text{if } M>1, \end{cases}$$

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

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

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

4.A.14 Youden's J Statistic

The *Youden's J Statistic* J_θ, *Youden's Index*, or *(Bookmaker) Informedness* was introduced by Youden [193] to capture the performance of a diagnostic test as a single statistic. It incorporates both the True Positive Rate and the True Negative Rate, which are discussed in Sections 4.A.5 and 4.A.6, respectively. Youden's J Statistic shows how well the model is able to correctly predict both the positive as the negative observations.

Definition and distribution

The Youden's J Statistic is commonly defined as

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

By using Equations (4.5) and (4.6), which provide the definitions of TPR_θ and TNR_θ in terms of TP_θ, the definition of J_θ can be reformulated as

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

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

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

with range:

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

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

$$\mathbb{E}[\mathrm{J}_\theta] = \mathbb{E}\left[X_\theta\left(\frac{M}{P\,(M-P)}, -\frac{\lfloor M\cdot\theta\rceil}{M-P}\right)\right] \overset{(\square)}{=} \frac{M}{P\,(M-P)} \cdot \mathbb{E}[\mathrm{TP}_\theta] - \frac{\lfloor M\cdot\theta\rceil}{M-P}$$

$$= 0.$$

Optimal baselines

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

$$\min_{\theta\in[0,1]} \{\mathbb{E}[\mathrm{J}_\theta]\} = 0,$$

$$\max_{\theta\in[0,1]} \{\mathbb{E}[\mathrm{J}_\theta]\} = 0,$$

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

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

This also holds for the discrete forms $\theta^*_{\min}$ and $\theta^*_{\max}$ of the optimizers:

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

4.A.15 Markedness

The *Markedness* MK_θ or *deltaP* is a performance measure that is mostly used in linguistics and social sciences. It combines both the Positive Predictive Value and the Negative Predictive Value. These two measures are discussed in Sections 4.A.9 and 4.A.10, respectively. The Markedness indicates how cautious the model is in predicting observations as positive and also how cautious it is in predicting them as negative.

Definition and distribution

The Markedness is commonly defined as

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

This definition of MK_θ can be reformulated in terms of TP_θ by using Equations (4.9) and (4.10):

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

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

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

with range:

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

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 calculated:

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

Optimal baselines

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

$$\min_{\theta \in [1/(2M), 1-1/(2M))} \{\mathbb{E}[\mathrm{MK}_\theta]\} = 0,$$

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

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

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

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

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

4.A.16 Accuracy

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

Definition and distribution

The Accuracy is commonly defined as

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

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

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

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

$$\text{Acc}_\theta = X_\theta\left(\frac{2}{M}, \frac{M - P - \lfloor M \cdot \theta \rceil}{M}\right) \sim f_{X_\theta}\left(\frac{2}{M}, \frac{M - P - \lfloor M \cdot \theta \rceil}{M}\right), \tag{4.12}$$

with range:

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

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:

$$\begin{aligned}\mathbb{E}[\mathrm{Acc}_\theta] &= \mathbb{E}\left[X_\theta\left(\frac{2}{M}, \frac{M - P - \lfloor M \cdot \theta \rceil}{M}\right)\right] \\ &\overset{(\square)}{=} \frac{2}{M} \cdot \mathbb{E}[\mathrm{TP}_\theta] + \frac{M - P - \lfloor M \cdot \theta \rceil}{M} \\ &= \frac{(M - \lfloor M \cdot \theta \rceil)\,(M - P) + \lfloor M \cdot \theta \rceil \cdot P}{M^2} \\ &= \frac{(1 - \theta^*)\,(M - P) + \theta^* \cdot P}{M}. \end{aligned} \tag{4.13}$$

Optimal baselines

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

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

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

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

It does not depend on t, but whether the derivative is positive or negative depends on P and M. Whenever $P > \frac{M}{2}$, then f is strictly increasing for all t in its domain. If $P < \frac{M}{2}$, then f is strictly decreasing. When $P = \frac{M}{2}$, f is constant. Consequently, the same holds for $\mathbb{E}[\mathrm{Acc}_\theta]$ given in Equation (4.13). This is because the term $\lfloor M \cdot \theta \rceil / M$ is non-decreasing in θ. Thus, the extreme values of the expectation of Acc_θ are given by

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

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

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

$$\theta_{\min} \in \underset{\theta\in[0,1]}{\arg\min} \{\mathbb{E}[\mathrm{Acc}_\theta]\} = \begin{cases} \left[1-\frac{1}{2M},1\right] & \text{if } P < \frac{M}{2} \\ [0,1] & \text{if } P = \frac{M}{2} \\ \left[0,\frac{1}{2M}\right) & \text{if } P > \frac{M}{2}, \end{cases} \tag{4.14}$$

$$\theta_{\max} \in \underset{\theta\in[0,1]}{\arg\max} \{\mathbb{E}[\mathrm{Acc}_\theta]\} = \begin{cases} \left[0,\frac{1}{2M}\right) & \text{if } P < \frac{M}{2} \\ [0,1] & \text{if } P = \frac{M}{2} \\ \left[1-\frac{1}{2M},1\right] & \text{if } P > \frac{M}{2}. \end{cases} \tag{4.15}$$

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

$$\theta^*_{\min} \in \underset{\theta^*\in\Theta^*}{\arg\min} \{\mathbb{E}[\mathrm{Acc}_{\theta^*}]\} = \begin{cases} \{1\} & \text{if } P < \frac{M}{2} \\ \Theta^* & \text{if } P = \frac{M}{2} \\ \{0\} & \text{if } P > \frac{M}{2}, \end{cases} \tag{4.16}$$

$$\theta^*_{\max} \in \underset{\theta^*\in\Theta^*}{\arg\max} \{\mathbb{E}[\mathrm{Acc}_{\theta^*}]\} = \begin{cases} \{0\} & \text{if } P < \frac{M}{2} \\ \Theta^* & \text{if } P = \frac{M}{2} \\ \{1\} & \text{if } P > \frac{M}{2}, \end{cases} \tag{4.17}$$

respectively.

4.A.17 Balanced Accuracy

The *Balanced Accuracy* BAcc_θ is the mean of the True Positive Rate and True Negative Rate, which are discussed in Sections 4.A.5 and 4.A.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 Balanced Accuracy is commonly defined as

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

By using Equations (4.5) and (4.6), this can be reformulated as

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

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

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

with range:

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

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:

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

Optimal baselines

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

$$\min_{\theta\in[0,1]}\{\mathbb{E}[\mathrm{BAcc}_\theta]\} = \frac{1}{2},$$

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

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

$$\theta_{\min} \in \arg\min_{\theta \in [0,1]} \{\mathbb{E}[\mathrm{BAcc}_\theta]\} = [0,1],$$

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

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

$$\theta^*_{\min} \in \arg\min_{\theta^* \in \Theta^*} \{\mathbb{E}[\mathrm{Acc}_{\theta^*}]\} = \Theta^*,$$

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

respectively.

4.A.18 Matthews Correlation Coefficient

The *Matthews Correlation Coefficient* MCC_θ was established by Matthews [113]. However, its definition is identical to that of the Yule phi coefficient, which was introduced by Yule [194]. The performance measure can be seen as the correlation coefficient between the actual and predicted classes. Hence, it is one of the few measures that lies in $[-1,1]$ instead of $[0,1]$.

Definition and distribution

The Matthews Correlation Coefficient is commonly defined as

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

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

$$\mathrm{MCC}_\theta = \frac{M \cdot \mathrm{TP}_\theta - P \cdot \lfloor M \cdot \theta \rceil}{\sqrt{\lfloor M \cdot \theta \rceil \cdot P \,(M - P)\,(M - \lfloor M \cdot \theta \rceil)}}. \tag{4.18}$$

As Table 4.2 shows, the assumptions $P > 0$, $N > 0$, $\hat{P} := \lfloor M \cdot \theta \rceil > 0$, and $\hat{N} := M - \lfloor M \cdot \theta \rceil > 0$ must hold. If one of these assumptions is violated, then the denominator in Equation (4.18) 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 (4.18):

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

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

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

with range:

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

4

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 (□):

$$\mathbb{E}[\mathrm{MCC}_\theta] = \mathbb{E}\left[X_\theta \left(\frac{M}{C(M, P, \theta)}, \frac{-P \cdot \lfloor M \cdot \theta \rceil}{C(M, P, \theta)} \right) \right] \overset{(\square)}{=} \frac{M}{C(M, P, \theta)} \cdot \mathbb{E}[\mathrm{TP}_\theta] - \frac{P \cdot \lfloor M \cdot \theta \rceil}{C(M, P, \theta)} = 0.$$

Optimal baselines

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

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

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

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

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

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

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

4.A.19 Cohen's Kappa

Cohen's Kappa κ_θ is a less straightforward performance measure than the other measures that we discuss in this research. It is used to quantify the inter-rater reliability for two raters of categorical observations [98]. 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 Cohen's Kappa, here we choose the following:

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

with P_o^θ the Accuracy Acc_θ as defined in Section 4.A.16 and P_e^θ the probability that the shuffle approach assigns the true label by chance. These two values can be expressed in terms of the base measures as follows:

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

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

By using Equations (4.12) and (B1) to (B4) the above can be rewritten as

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

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

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

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

This simplifies to

$$\frac{\lfloor M \cdot \theta \rceil}{M} \neq \frac{P}{2P - M}. \tag{4.19}$$

The left-hand side is by definition in the interval $[0, 1]$. For the right-hand side to be in that interval, we firstly need $P/(2P-M) \geq 0$. Since $P \geq 0$, that means $2P - M > 0$, and hence, $P > \frac{M}{2}$. Secondly, $P/(2P - M) \leq 1$. Since we know $P > \frac{M}{2}$, we obtain $P \geq M$. This inequality reduces to $P = M$, because P is always at most M. Whenever $P = M$, then Equation (4.19) 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 Cohen's Kappa, we obtain:

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

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

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

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

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

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

with range:

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

Expectation

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

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

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

$$= 0.$$

Optimal baselines

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

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

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

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

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

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

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

4.A.20 Fowlkes-Mallows Index

The *Fowlkes-Mallows Index* FM_θ or *G-mean 1* was introduced by [51] as a way to calculate the similarity between two clusterings. It is the geometric average between the True Positive Rate (TPR_θ) and Positive Predictive Value (PPV_θ), which are discussed in Sections 4.A.5 and 4.A.9, respectively. It offers a balance between correctly predicting the actual positive observations (TPR_θ) and being cautious in predicting observations as positive (PPV_θ).

Definition and distribution

The Fowlkes-Mallows Index is commonly defined as

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

By using the definitions of TPR_θ and PPV_θ in terms of TP_θ in, respectively, Equations (4.5) and (4.9), 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:

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

Expectation

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

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

Optimal baselines

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

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

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

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

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

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

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

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

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

4.A.21 G-mean 2

The *G-mean 2* $G_\theta^{(2)}$ was established by [94]. This performance measure is the geometric average between the True Positive Rate (TPR_θ) and True Negative Rate (TNR_θ), which we discuss in Sections 4.A.5 and 4.A.6, respectively. Hence, it balances correctly predicting the positive observations and correctly predicting the negative observations.

Definition and distribution

The G-mean 2 is defined as

$$G_\theta^{(2)} = \sqrt{TPR_\theta \cdot TNR_\theta}.$$

Since TPR_θ needs the assumption $P > 0$ and TNR_θ needs $N := M - P > 0$, we have these restrictions also for $G_\theta^{(2)}$. Consequently, $M > 1$. Now, by using the definitions of TPR_θ and TNR_θ in terms of TP_θ in, respectively, Equations (4.5) and (4.6), we obtain:

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

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

Expectation

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

moment of $\mathrm{G}_\theta^{(2)}$:

$$\mathbb{E}\left[\left(\mathrm{G}_\theta^{(2)}\right)^2\right] = \frac{M-P-\lfloor M\cdot\theta\rceil}{P\,(M-P)}\cdot\mathbb{E}[\mathrm{TP}_\theta] + \frac{1}{P\,(M-P)}\cdot\mathbb{E}[\mathrm{TP}_\theta^2]$$

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

$$= \frac{(M-P-\lfloor M\cdot\theta\rceil)\cdot\lfloor M\cdot\theta\rceil}{M\,(M-P)} + \frac{\frac{\lfloor M\cdot\theta\rceil(M-\lfloor M\cdot\theta\rceil)P(M-P)}{M^2(M-1)} + \left(\frac{\lfloor M\cdot\theta\rceil}{M}\cdot P\right)^2}{P\,(M-P)}$$

$$= \frac{\lfloor M\cdot\theta\rceil\cdot(M-\lfloor M\cdot\theta\rceil)}{M(M-1)} = \theta^*\cdot(1-\theta^*)\cdot\frac{M}{M-1}.$$

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

Optimal baselines

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

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

This means that

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

Therefore, whenever $\theta^*\in\{0,1\}$, then $\mathbb{E}[\mathrm{G}_\theta^{(2)}]\leq 0$. Since $\mathrm{G}_\theta^{(2)}\geq 0$, it must hold that $\mathbb{E}[\mathrm{G}_\theta^{(2)}]=0$. Hence, the set $\{0,1\}$ contains minimizers for $\mathbb{E}[\mathrm{G}_\theta^{(2)}]$. The continuous version of this set is the interval $[0,1/(2M))\cup[1-1/(2M),1]$.

To show that this interval contains the only possible values for the minimizers, consider the definition for the expectation of $\mathrm{G}_\theta^{(2)}$:

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

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

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

which is strictly positive in both cases. Hence, there is at least one term in the summation in the definition of $\mathbb{E}\left[\mathrm{G}_\theta^{(2)}\right]$ that is larger than 0, thus the expectation is strictly positive for $1/(2M) \leq \theta < 1 - 1/(2M)$. Consequently, the minimization values $\theta_{\min} \in [0, 1]$ are

$$\theta_{\min} \in \arg\min_{\theta\in[0,1]} \left\{\mathbb{E}[\mathrm{G}_\theta^{(2)}]\right\} = \left[0, \frac{1}{2M}\right) \cup \left[1-\frac{1}{2M}, 1\right].$$

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

$$\theta^*_{\min} \in \arg\min_{\theta^*\in\Theta^*} \left\{\mathbb{E}[\mathrm{G}_\theta^{(2)}]\right\} = \{0, 1\}.$$

4.A.22 Prevalence Threshold (PT)

A relatively new performance measure named *Prevalence Threshold* (PT_θ) was introduced by [11]. We could not find many articles that use this measure, but it is included for completeness. However, this performance measure has an inherent problem that eliminates the possibility to determine all statistics.

Definition and distribution

The *Prevalence Threshold* PT_θ is commonly defined as

$$\mathrm{PT}_\theta = \frac{\sqrt{\mathrm{TPR}_\theta\cdot\mathrm{FPR}_\theta}-\mathrm{FPR}_\theta}{\mathrm{TPR}_\theta-\mathrm{FPR}_\theta}.$$

By using the definitions of TPR_θ and FPR_θ in terms of TP_θ (see Equations (4.5) and (4.7)), we obtain:

$$\mathrm{PT}_\theta = \frac{\sqrt{P \cdot (M-P) \cdot \mathrm{TP}_\theta \cdot (\lfloor M \cdot \theta \rceil - \mathrm{TP}_\theta)} - P(\lfloor M \cdot \theta \rceil - \mathrm{TP}_\theta)}{M \cdot \mathrm{TP}_\theta - P \cdot \lfloor M \cdot \theta \rceil}. \tag{4.20}$$

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

Equation (4.20) 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. The number of True Positives TP_θ can attain the value $(\lfloor M \cdot \theta \rceil / M) \cdot P = \theta^* \cdot P$ whenever the latter is also an integer. For example, this always happens for $\theta^* \in \{0, 1\}$. But even when $\theta^* \in \Theta^* \setminus \{0, 1\}$, PT_θ is still only safe to use when M and P are coprime, i.e., when the only positive integer that is a divisor of both of them is 1. Otherwise, there are always values of $\theta^* \in \Theta^* \setminus \{0, 1\}$ that cause $\theta^* \cdot P$ to be an integer and therefore PT_θ to be undefined when TP_θ attains that value.

One solution would be to say $\mathrm{PT}_\theta := c$, $c \in [0, 1]$, whenever both the numerator and denominator are zero. However, this c is arbitrary and directly influences the 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 the Prevalence Threshold PT_θ.

4.A.23 Threat Score (TS) / Critical Success Index (CSI)

The *Threat Score* [125] TS_θ or *Critical Success Index* [151] is a performance measure that is used for evaluation of forecasting binary weather events: it either happens in a specific location or it does not. It was already used in 1884 to evaluate the prediction of tornadoes [151]. The Threat Score is the ratio of successful event forecasts (TP_θ) to the total number of positive predictions ($\mathrm{TP}_\theta + \mathrm{FP}_\theta$) and the number of events that were missed (FN_θ).

Definition and distribution

The Threat Score is thus defined as

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

By using Equations (B2) and (B3), this definition can be reformulated as

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

Note that TS_θ is well-defined whenever $P > 0$. The definition of TS_θ is not linear in TP_θ, and so there are no $a, b \in \mathbb{R}$ such that we can write the definition as $X_\theta\,(a, b)$.

4

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

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

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

Optimal baselines

Although no explicit formula can be given for the expectation, we are able to calculate the extreme values of the expectation and the corresponding optimizers.

Minimal baseline Firstly, we show that $\theta_{\min} \in [0, \frac{1}{2M})$ constitutes a minimum and that there are no θ outside this interval also yielding this minimum. To this end,

$$\mathbb{E}[\mathrm{TS}_{\theta_{\min}}] = \sum_{k \in \mathcal{D}(\mathrm{TS}_{\theta_{\min}})} \frac{k}{P + 0 - k} \cdot \mathbb{P}(\mathrm{TS}_{\theta_{\min}} = k) = 0,$$

because $\mathcal{D}(\mathrm{TS}_{\theta_{\min}}) = \{0\}$. This is the lowest possible value, since TS_θ is a non-negative performance measure, and hence, $\mathbb{E}[\mathrm{TS}_\theta] \geq 0$ for any $\theta \in [0,1]$. Now, let $\theta' \geq \frac{1}{2M}$, then there exists a $k' > 0$ such that $\mathbb{P}(\mathrm{TP}_{\theta'} = k') > 0$. Consequently, $\mathbb{E}[\mathrm{TS}_{\theta'}] > 0$ and this means the interval $[0, \frac{1}{2M})$ contains the only values that constitute the minimum. In summary,

$$\min_{\theta \in [0,1]} \{\mathbb{E}[\mathrm{TS}_\theta]\} = 0,$$

$$\theta_{\min} \in \arg\min_{\theta \in [0,1]} \{\mathbb{E}[\mathrm{TS}_\theta]\} = \left[0, \frac{1}{2M}\right).$$

Since $\theta^*_{\min}$ is the discretization of $\theta_{\min}$ it corresponds to 0. More precisely:

$$\theta^*_{\min} \in \arg\min_{\theta^* \in \Theta^*} \{\mathbb{E}[\mathrm{TS}_{\theta^*}]\} = \{0\}.$$

Maximal baseline Secondly, to determine the maximum of $\mathbb{E}[\mathrm{TS}_\theta]$ and the corresponding parameter $\theta_{\max}$, we determine an upper bound for the expectation, show that this value is attained for a specific interval and that there is no θ outside this interval also yielding this value. To do this, assume that $\lfloor M \cdot \theta \rceil > 0$. This makes sense, because $\lfloor M \cdot \theta \rceil = 0$ implies $\theta < 1/(2M)$ and such a θ would yield the minimum 0. Now,

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

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

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

because $\mathbb{P}(\mathrm{TP}_{\theta_{\max}} = P) = 1$. Hence, the upper bound is attained for $\theta_{\max} \in [1 - 1/(2M), 1]$, and thus, $\theta_{\max}$ is a maximizer.

Now, specifically for $P = 1$, we show that the interval of maximizers is actually $[1/(2M), 1]$. Thus, let $\theta \in [1/(2M), 1-1/(2M))$, then $0 < \lfloor M \cdot \theta \rceil < M$ and

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

which is exactly the upper bound $\mathbb{E}[\mathrm{TS}_{\theta_{\max}}] = P/M$ for $P = 1$.

Next, to show that the maximizers are only in $[1 - 1/(2M), 1]$ for $P > 1$, assume there is a $\theta' < 1 - \frac{1}{2M}$ that also yields the maximum. Hence, there is a $k' \in \mathcal{D}(\mathrm{TP}_{\theta'})$ with $0 < k' < P$ such that $\mathbb{P}(\mathrm{TP}_{\theta'} = k')$. This means that

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

$$= \frac{k'}{P+\lfloor M\cdot\theta'\rceil - k'}\cdot\mathbb{P}(\mathrm{TP}_{\theta'}=k') + \sum_{k\in\mathcal{D}(\mathrm{TP}_{\theta'})\setminus\{k'\}} \frac{k}{P+\lfloor M\cdot\theta'\rceil - k}\cdot\mathbb{P}(\mathrm{TP}_{\theta'}=k)$$

$$\leq \frac{k'}{P+\lfloor M\cdot\theta'\rceil - (P-1)}\cdot\mathbb{P}(\mathrm{TP}_{\theta'}=k') + \sum_{k\in\mathcal{D}(\mathrm{TP}_{\theta'})\setminus\{k'\}} \frac{k}{P+\lfloor M\cdot\theta'\rceil - P}\cdot\mathbb{P}(\mathrm{TP}_{\theta'}=k)$$

$$= \frac{k'}{\lfloor M\cdot\theta'\rceil+1}\mathbb{P}(\mathrm{TP}_{\theta'}=k') + \sum_{k\in\mathcal{D}(\mathrm{TP}_{\theta'})\setminus\{k'\}} \frac{k}{\lfloor M\cdot\theta'\rceil}\mathbb{P}(\mathrm{TP}_{\theta'}=k)$$

$$< \frac{k'}{\lfloor M\cdot\theta'\rceil}\cdot\mathbb{P}(\mathrm{TP}_{\theta'}=k') + \sum_{k\in\mathcal{D}(\mathrm{TP}_{\theta'})\setminus\{k'\}} \frac{k}{\lfloor M\cdot\theta'\rceil}\cdot\mathbb{P}(\mathrm{TP}_{\theta'}=k)$$

$$= \frac{1}{\lfloor M\cdot\theta'\rceil}\sum_{k\in\mathcal{D}(\mathrm{TP}_{\theta'})} k\cdot\mathbb{P}(\mathrm{TP}_{\theta'}=k) = \frac{P}{M}.$$

Hence, there is a strict inequality $\mathbb{E}[\mathrm{TS}_{\theta'}] < \frac{P}{M}$ 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:

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

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

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

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

Chapter 5

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

Contents

Based on *Joris Pries, Etienne van de Bijl, Jan Klein, Sandjai Bhulai, and Rob van der Mei* (): "**The optimal input-independent baseline for binary classification: The Dutch Draw**". Accepted for publication in *Statistica Neerlandica*. [138]

Abstract

Before any binary classification model is taken into practice, it is important 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 of 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 recently proposed *Dutch Draw baseline* is the best *input-independent* classifier (independent of feature values) for all *positional-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.

5.1 Introduction

A *binary classification* model is trying 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 [130], detecting malicious intrusions [103] to determining if a runner is fatigued or not [24]. Whenever a classification model is developed for a practical application, it is important to validate the performance on a test set. However, 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. But it could be that the model actually did not learn anything and such a high accuracy can 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, Van de Bijl et al. [14] plead 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 e.g., 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 is proposed in [14] based on *Dutch Draw 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 *Dutch Draw baseline*. For most commonly used performance measures, a closed-form expression is given [14].

However, there are infinitely many ways to construct a baseline. 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 *permutation-optimality*. Furthermore, the order of the samples should not change the outcome of the performance measure (*positional-invariant*). This is not a strict condition, as most commonly used measures have this property. Under these restrictions, we prove that the *Dutch Draw baseline* is *permutation-optimal* out of all *input-independent* classifiers for any *positional-invariant* measure.

To summarize, in this chapter we:

- determine natural requirements for a general, informative and simple baseline;
- prove that the Dutch Draw 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 5.2. Next, in Section 5.3 we determine requirements for a general, simple and informative baseline. Furthermore, we formally define what optimality entails under these requirements. In Section 5.4, an alternative definition for the *Dutch Draw classifiers* is given, which is necessary for the main proof. In Section 5.5, we prove that the Dutch Draw baseline is optimal. Finally, Section 5.6 summarizes the general findings and discusses possible future research opportunities.

5.2 Preliminaries

Next, we introduce some concepts and notations to lay the foundation for the main proof. First, *binary classifiers* (Section 5.2.1) and *performance measures* (Section 5.2.2) for binary classification are discussed. Then, properties of *permutations* are examined in Section 5.2.3, which will play a crucial role in the proof of the main result.

5.2.1 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 $M \in \mathbb{N}_{>0}$ denotes the number of samples that are classified. This gives classifiers the ability to precisely predict k out of M samples positive. Note that a *single sample* classifier h can simply be extended to classify M samples *simultaneously* by applying the classifier for each sample individually:

$$h_M : ((x_1, \ldots, x_M), r) \mapsto (h(x_1, r), \ldots, h(x_M, r)) .$$

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.

Example of a binary classifier

An example of a binary classifier is a *coin toss*, where each sample is classified by throwing a coin and determining on which side it lands. Let $\theta \in [0, 1]$ be the probability that the coin lands head, and $1 - \theta$ for tails. Assuming that head and tails are classified by 1 and 0 respectively, we get:

$$h^{\text{single}}_{\text{coin}}(\cdot, r) := \begin{cases} 1 & \text{with probability } \theta, \\ 0 & \text{with probability } 1 - \theta. \end{cases}$$

Classifying M samples by repeatedly throwing coins can be achieved by:

$$h_{\text{coin}} : ((x_1, \ldots, x_M), r) \mapsto \left(h^{\text{single}}_{\text{coin}}(x_1, r), \ldots, h^{\text{single}}_{\text{coin}}(x_M, r) \right) .$$

5.2.2 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 $\mathbf{X} := (\mathbf{x}_1 \dots \mathbf{x}_M) \in \mathcal{X}^M$ be the combined feature values of M samples. Furthermore, let $\mathbf{Y} = (y_1, \dots, y_M)$ 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: $\mu(h_M(\mathbf{X}, r), \mathbf{Y})$.

Example of a performance measure

An example of a performance measure for binary classification is *accuracy* (μ_{acc}). It is defined as the total number of correctly classified samples divided by the total number of samples. For any $h_M(\mathbf{X}, r) = (\hat{y}_1, \dots, \hat{y}_M) \in \{0,1\}^M$ and $\mathbf{Y} = (y_1, \dots, y_M) \in \{0,1\}^M$, it holds that

$$\mu_{\text{acc}}\left(h_M(\mathbf{X}, r), \mathbf{Y}\right) = \frac{\sum_{i=1}^{M} \mathbb{1}_{\{\hat{y}_i = y_i\}}}{M}.$$

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* [174], 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* [174] 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

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

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.

5.2.3 Permutations

To determine which binary classifier is considered to be the 'best', we define *permutation-optimality* in Section 5.3.3, which uses *permutations* to define 'optimality'. In this section, we examine properties of permutations that are used in the main proof (see Section 5.5). A permutation is a *bijective* function from a set to itself [46]. This means that a permutation is not a reordered list; it is a function that determines where each element should be rearranged to.

Let S_M denote the set of all permutations of a set of size M, also called the *symmetric group*. More formally,

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

Example of symmetric group

Using the Cauchy one-line notation [34], all possible permutations of three elements are given by

$$\begin{pmatrix}1 & 2 & 3\end{pmatrix}, \quad \begin{pmatrix}1 & 3 & 2\end{pmatrix}, \quad \begin{pmatrix}2 & 1 & 3\end{pmatrix},$$
$$\begin{pmatrix}2 & 3 & 1\end{pmatrix}, \quad \begin{pmatrix}3 & 1 & 2\end{pmatrix}, \quad \begin{pmatrix}3 & 2 & 1\end{pmatrix}.$$

The permutation $\begin{pmatrix}2 & 3 & 1\end{pmatrix}$ sends the first element to the second position, the second element to the third position and the third element to the first position.

Sample-wise permutations

To apply permutations to a matrix, we discuss *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 $K \in \mathbb{N}_{>0}$ the number of features. This means that the matrix X is reordered by row.

Properties of permutations

Next, we outline some properties of S_M that are used in the proof of the main result. S_M is a group with the composition of functions as group operator (denoted by $\circ$), thus the group axioms must hold [10, 46]. This means that there exists an *identity element* $\mathrm{id} \in S_M$ such that for all $\pi \in S_M$:

$$\mathrm{id} \circ \pi = \pi = \pi \circ \mathrm{id}.$$

Furthermore, for every $\pi \in S_M$, there exists a unique *inverse element* $\pi^{-1} \in S_M$ such that

$$\pi \circ \pi^{-1} = \mathrm{id} = \pi^{-1} \circ \pi.$$

Thus, for each permutation, there exists an inverse permutation that reverses the change of order of the permutation, which is used in Section 5.5. As each inverse is unique and also contained in S_M, it follows that

$$\{\pi \in S_M\} = \{\pi^{-1} : \pi \in S_M\}, \tag{5.1}$$

which means that the set of all permutations is the same as the set of all inverses of these permutations. Thus, taking an expectation over all permutations in S_M is the same as taking the expectation over all inverse permutations of permutations in S_M. This is used in the proof of the main result in Section 5.5.

5.3 Essential conditions

To prove that the optimal Dutch Draw 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, *positional-invariant* measures, and *permutation-optimality*.

5.3.1 Input-independent classifier

Any binary classifier can be used as a baseline. However, any good standardized baseline should be *general*, *simple*, and *informative* [14]. 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 named *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. By this definition, an input-independent classifier is not dependent on feature values or even the feature domains. 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.

5

Example of an input-independent classifier

The *coin flip* (see Section 5.2.1) is by definition input-independent. The feature values have no influence on the probability distribution of the coin. Thus, for any $(x_1, \ldots, x_M) \in \mathcal{X}^M$,

$$\left(h_{\text{coin}}^{\text{single}}(x_1), \ldots, h_{\text{coin}}^{\text{single}}(x_M)\right) = \left(h_{\text{coin}}^{\text{single}}(\cdot), \ldots, h_{\text{coin}}^{\text{single}}(\cdot)\right).$$

5.3.2 Positional-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 *positional-invariant*. More formally, a measure μ is *positional-invariant* if for every permutation $\pi \in S_M$ and for all $h_M(\mathbf{X}), \mathbf{Y} \in \{0,1\}^M$ it holds that:

$$\mu\left(h_M(\mathbf{X}, r), \mathbf{Y}\right) = \mu(h_M(\mathbf{X}, r)_\pi, \mathbf{Y}_\pi). \tag{5.2}$$

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 *positional-invariant*. Most commonly used measures are a function of these four measures [163], making them also *positional-invariant*.

Example of a non-positional-invariant measure

Nonetheless, it is possible to define measures that are not *positional-invariant*. For example, take the measure

$$\lambda : \{0,1\}^M \times \{0,1\}^M \to \mathbb{R}, (a = (a_1, \ldots, a_M), b) \mapsto a_1,$$

which is dependent on the first position of the prediction, as

$$\lambda\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right) = 0,$$

$$\lambda\left(\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\right) = 1.$$

5

5.3.3 Defining optimality

To find the 'optimal' baseline, it is first essential to specify what 'optimality' entails.

Optimal classifier

A binary classifier does not need to have a deterministic outcome. Thus, due to *stochasticity*, we consider a classifier to be *optimal* if it minimizes/-maximizes the expected performance out of all considered binary classifiers (denoted by $\tilde{\mathcal{H}}_M$). This means, that we have to average the performance over all $r \in \mathbb{R}$. Whether optimization means minimization or maximization depends on the objective of the problem. So:

$$h_M^{\min} \in \underset{h_M \in \tilde{\mathcal{H}}_M}{\arg\min} \left\{ \mathbb{E}_{r \in \mathbb{R}} \left[\mu(h_M(\mathbf{X}, r), \mathbf{Y}) \right] \right\}, \tag{5.3}$$

$$h_M^{\max} \in \underset{h_M \in \tilde{\mathcal{H}}_M}{\arg\max} \left\{ \mathbb{E}_{r \in \mathbb{R}} \left[\mu(h_M(\mathbf{X}, r), \mathbf{Y}) \right] \right\}. \tag{5.4}$$

For example, when the goal is to maximize the accuracy, then $h_M^{\max}$ is an optimal baseline out of all other baselines in $\tilde{\mathcal{H}}_M$. Note that there could be multiple different optimal baselines.

Trivial optimal solution

However, this definition of 'optimality' leads to a trivial optimal solution, when we consider all *input-independent* classifiers ($\tilde{\mathcal{H}}_M = \mathcal{H}_M^{i.i.}$). Take the deterministic classifier

$$\tilde{h}_M^{\max}(\cdot, r) := \hat{\mathbf{Y}}_{\max} \in \underset{\hat{\mathbf{Y}} \in \{0,1\}^M}{\arg\max}\, \mu(\hat{\mathbf{Y}}, \mathbf{Y}),$$

which always predicts a vector $\hat{\mathbf{Y}}_{\max}$ that maximizes the measure μ. Note that $\tilde{h}_M^{\max}$ is clearly *input-independent* (see Section 5.3.1), thus $\tilde{h}_M^{\max} \in \mathcal{H}_M^{i.i.}$. Furthermore, it holds that

$$\begin{aligned}\max_{h_M \in \tilde{\mathcal{H}}_M} \left\{\mathbb{E}_{h_M(\mathbf{X})}\left[\mu(h_M(\mathbf{X}), \mathbf{Y})\right]\right\} &\leq \max_{\hat{\mathbf{Y}} \in \{0,1\}^M} \mu(\hat{\mathbf{Y}}, \mathbf{Y}) \\ &= \mathbb{E}_{\tilde{h}_M^{\max}(r)}\left[\mu(\tilde{h}_M^{\max}(r), \mathbf{Y})\right].\end{aligned}$$

In other words, the expected performance of $\tilde{h}_M^{\max}$ is always higher or equal compared to any other classifier. Thus, $\tilde{h}_M^{\max}$ is considered to be *optimal* (see Equation (5.4)). The same holds for minimization with

$$\tilde{h}_M^{\min}(r) := \hat{\mathbf{Y}}_{\min} \in \underset{\hat{\mathbf{Y}} \in \{0,1\}^M}{\arg\min}\, \mu(\hat{\mathbf{Y}}, \mathbf{Y}).$$

Essentially, a perfect prediction can always be made by an *input-independent* classifier, using the *actual* labels and the performance measure. Consider for example the commonly used performance measure: *accuracy*, which is maximized if the prediction $\hat{\mathbf{Y}} = \mathbf{Y}$. A classifier $\tilde{h}_M^{\max}$ that always predicts $\mathbf{Y}$, is thus *optimal* for these given labels. This shows that an extension to the definition of 'optimality' should be considered.

Permutation-optimality

The *optimal* property (see Equations (5.3) and (5.4)) is not really insightful when we consider all deterministic classifiers, as the perfect prediction is always made by one of them. Similarly, a broken clock gives the correct time

twice a day, but should not be used to determine the time. Therefore, we introduce a new optimality condition named *permutation-optimality*.

It is often assumed that the test set is randomly shuffled. Therefore, we introduce the notion of *permutation-optimality*. Instead of being optimal for the distinct order that the feature values and corresponding labels are given in, now all permutations of the samples are considered. A classifier is *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,

$$h_M^{\min} \in \underset{h_M \in \tilde{\mathcal{H}}_M}{\arg\min} \left\{ \mathbb{E}_{\pi \sim \mathcal{U}(S_M)} \left[\mathbb{E}_{r \in \mathbb{R}} \left[\mu(h_M(\mathbf{X}_\pi, r), \mathbf{Y}_\pi) \right] \right] \right\}, \tag{5.5}$$

$$h_M^{\max} \in \underset{h_M \in \tilde{\mathcal{H}}_M}{\arg\max} \left\{ \mathbb{E}_{\pi \sim \mathcal{U}(S_M)} \left[\mathbb{E}_{r \in \mathbb{R}} \left[\mu(h_M(\mathbf{X}_\pi, r), \mathbf{Y}_\pi) \right] \right] \right\}. \tag{5.6}$$

5.4 The Dutch Draw

A Dutch Draw classifier is defined in [14] for $\theta \in [0, 1]$, as

$$\sigma_\theta(\mathbf{X}, \cdot) := (\mathbf{1}_E(i))_{i \in \{1, \dots M\}} \text{ with } E \subseteq \{1, \dots M\} \text{ drawn u.a.r. such that } |E| = \lfloor M \cdot \theta \rceil. \tag{5.7}$$

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 Dutch Draw classifiers are *input-independent*.

5.4.1 Alternative definition

Instead of the definition in Equation (5.7), we introduce an alternative definition for the Dutch Draw 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

$$\mathcal{Y}_j := \left\{ \hat{\mathbf{Y}} = (y_1, \dots, y_M) \in \{0, 1\}^M \text{ s.t. } \sum_{i=1}^M y_i = j \right\}. \tag{5.8}$$

In other words, $\mathcal{Y}_j$ contains all binary vectors of length M with exactly j ones and $M - j$ zeros. For example, for $M = 4$ it holds that

$$\mathcal{Y}_0 = \{(0,0,0,0)\},$$

$$\mathcal{Y}_1 = \{(0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0)\},$$

$$\mathcal{Y}_2 = \{(0,0,1,1),(0,1,0,1),(0,1,1,0),(1,0,0,1),(1,0,1,0),(1,1,0,0)\},$$

$$\mathcal{Y}_3 = \{(0,1,1,1),(1,0,1,1),(1,1,0,1),(1,1,1,0)\},$$

$$\mathcal{Y}_4 = \{(1,1,1,1)\}.$$

A Dutch Draw 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 Dutch Draw classifier σ_θ can be rewritten as

$$\sigma_\theta(\mathbf{X}, \cdot) := \hat{\mathbf{Y}} \text{ with } \hat{\mathbf{Y}} \sim \mathcal{U}\left(\mathcal{Y}_{\lfloor M \cdot \theta \rceil}\right). \tag{5.9}$$

Put differently, a Dutch Draw 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.

5.4.2 Input-independence

Next, we discuss why all Dutch Draw classifiers are *input-independent* (see Section 5.3.1). Note that a Dutch Draw classifier σ_θ is independent of feature values, as it is only dependent on θ and M, see Equation (5.9). In other words, any Dutch Draw 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.

5.4.3 Optimal Dutch Draw classifier

The optimal Dutch Draw classifier $\sigma_{\theta_{\text{opt}}}$ is determined by minimizing/maximizing the expected performance for the parameter θ out of all allowed parameter values Θ [14]. Note that some measures are undefined for certain predictions, thus Θ is not always equal to $[0,1]$. Take e.g., the measure

precision [174], 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 5.2.2, all undefined cases are resolved and $\Theta = [0, 1]$ always holds.

Using the alternative definition of the Dutch Draw classifier (see Equation (5.9)), we obtain:

$$\theta^*_{\min} \in \arg\min_{\theta \in [0,1]} \left\{ \mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}(\mathcal{Y}_{\lfloor M \cdot \theta \rceil})} \left[\mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right] \right\}, \tag{5.10}$$

$$\theta^*_{\max} \in \arg\max_{\theta \in [0,1]} \left\{ \mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}(\mathcal{Y}_{\lfloor M \cdot \theta \rceil})} \left[\mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right] \right\}. \tag{5.11}$$

Depending on the objective, either $\sigma_{\theta^*_{\min}}$ or $\sigma_{\theta^*_{\max}}$ is an optimal Dutch Draw classifier.

5.5 Theorem and proof

After defining *input-independence* (Section 5.3.1), *positional-invariance* (Section 5.3.2), *permutation-optimality* (Section 5.3.3), and introducing an alternative formulation for the Dutch Draw classifier, all ingredients for the following theorem are present.

Theorem 5.5.1 (Main result)**.** *The optimal Dutch Draw classifier* $\sigma_{\theta_{opt}}$ *is* permutation-optimal *out of all* input-independent *classifiers* ($\mathcal{H}^{i.i.}_M$)*, for any* positional-invariant *measure* μ*. In other words:*

$$\sigma_{\theta^*_{min}} \in \arg\min_{h_M \in \mathcal{H}^{i.i.}_M} \left\{ \mathbb{E}_{\pi \sim \mathcal{U}(S_M)} \left[\mathbb{E}_{r \in \mathbb{R}} \left[\mu(h_M(\mathbf{X}_\pi, r), \mathbf{Y}_\pi) \right] \right] \right\}, \tag{5.12}$$

$$\sigma_{\theta^*_{max}} \in \arg\max_{h_M \in \mathcal{H}^{i.i.}_M} \left\{ \mathbb{E}_{\pi \sim \mathcal{U}(S_M)} \left[\mathbb{E}_{r \in \mathbb{R}} \left[\mu(h_M(\mathbf{X}_\pi, r), \mathbf{Y}_\pi) \right] \right] \right\}. \tag{5.13}$$

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

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 (*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 (*positional-invariance*). The optimal Dutch Draw classifier is determined by optimizing the number of predicted zeros and ones, which makes this baseline *permutation-optimal*.

Proof. Let $h_M \in \mathcal{H}_M^{i.i.}$ be an *input-independent* classifier and let μ be a *positional-invariant* measure, the classifier is *permutation-optimal* if it minimizes/maximizes the expected performance under a random permutation of the test set out of all *input-independent* classifiers (see Equations (5.5) and (5.6)).

For any *input-independent* classifier h_M, it holds that

$$\mathbb{E}_{r\in\mathbb{R}}\left[\mu(h_M(\mathbf{X}_\pi, r), \mathbf{Y}_\pi)\right] = \mathbb{E}_{r\in\mathbb{R}}\left[\mu(h_M(r), \mathbf{Y}_\pi)\right]. \tag{5.14}$$

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 (5.14) can be written out by:

$$\mathbb{E}_{r\in\mathbb{R}}\left[\mu(h_M(r), \mathbf{Y}_\pi)\right] = \sum_{i=1}^{2^M} \mathbb{P}(h_M(\cdot) = \hat{\mathbf{Y}}_i) \cdot \mu(\hat{\mathbf{Y}}_i, \mathbf{Y}_\pi). \tag{5.15}$$

As we need to proof *permutation-optimality*, we have to take the expectation of Equation (5.15) over all permutations. Using linearity of expectation gives:

$$\begin{aligned}\mathbb{E}_{\pi\sim\mathcal{U}(S_M)}&\left[\sum_{i=1}^{2^M} \mathbb{P}(h_M(\cdot) = \hat{\mathbf{Y}}_i) \cdot \mu(\hat{\mathbf{Y}}_i, \mathbf{Y}_\pi)\right] \\ &= \sum_{i=1}^{2^M} \mathbb{P}(h_M(\cdot) = \hat{\mathbf{Y}}_i) \cdot \mathbb{E}_{\pi\sim\mathcal{U}(S_M)}\left[\mu(\hat{\mathbf{Y}}_i, \mathbf{Y}_\pi)\right].\end{aligned} \tag{5.16}$$

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

The measure μ is *positional-invariant*, thus using Equation (5.2) gives

$$\mu(\hat{\mathbf{Y}}_i, \mathbf{Y}_\pi) = \mu((\hat{\mathbf{Y}}_i)_{\pi^{-1}}, (\mathbf{Y}_\pi)_{\pi^{-1}}) = \mu((\hat{\mathbf{Y}}_i)_{\pi^{-1}}, \mathbf{Y}). \tag{5.17}$$

Applying a permutation does not change a *positional-invariant* measure μ. In this case, we apply the inverse permutation π^{-1} to retrieve $\mathbf{Y}$.

Because of Equation (5.17), it therefore also holds that

$$\mathbb{E}_{\pi\sim\mathcal{U}(S_M)}\left[\mu(\hat{\mathbf{Y}}_i, \mathbf{Y}_\pi)\right] = \mathbb{E}_{\pi\sim\mathcal{U}(S_M)}\left[\mu((\hat{\mathbf{Y}}_i)_{\pi^{-1}}, \mathbf{Y})\right]. \tag{5.18}$$

Equation (5.1) shows that the set of all *inverse permutations* is the same as the set of all *permutations*. 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,

$$\begin{aligned}
\mathbb{E}_{\pi\sim\mathcal{U}(S_M)}\left[\mu((\hat{\mathbf{Y}}_i)_{\pi^{-1}}, \mathbf{Y})\right] &= \sum_{s\in S_M}\left(\mu((\hat{\mathbf{Y}}_i)_{s^{-1}}, \mathbf{Y}) \cdot \mathbb{P}(\pi = s)\right) \\
&= \sum_{s\in S_M}\left(\mu((\hat{\mathbf{Y}}_i)_{s^{-1}}, \mathbf{Y}) \cdot \mathbb{P}(\pi = s^{-1})\right) \\
&= \mathbb{E}_{\pi\sim\mathcal{U}(S_M)}\left[\mu((\hat{\mathbf{Y}}_i)_{\pi}, \mathbf{Y})\right]. \qquad (5.19)
\end{aligned}$$

Thus, π^{-1} can be replaced with π in Equation (5.18).

Recall that $\mathcal{Y}_j$ is the set of all binary vectors of length M with j ones (see Equation (5.8)). 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

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

and

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

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

$$\mathbb{E}_{\pi\sim\mathcal{U}(S_M)}\left[\mu((\hat{\mathbf{Y}}_i)_\pi, \mathbf{Y})\right] = \mathbb{E}_{\hat{\mathbf{Y}}\sim\mathcal{U}\left(\mathcal{Y}_{|\hat{\mathbf{Y}}_i|}\right)}\left[\mu(\hat{\mathbf{Y}}, \mathbf{Y})\right]. \tag{5.20}$$

Using Equations (5.18) to (5.20) in combination with Equation (5.16) 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 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 minimizing/maximizing over all possible values of j. Thus,

$$\mathbb{E}_{\hat{\mathbf{Y}}\sim\mathcal{U}(\mathcal{Y}_j)}\left[\mu(\hat{\mathbf{Y}}, \mathbf{Y})\right] \geq \min_{j'\in\{0,\ldots,M\}} \mathbb{E}_{\hat{\mathbf{Y}}\sim\mathcal{U}(\mathcal{Y}_{j'})}\left[\mu(\hat{\mathbf{Y}}, \mathbf{Y})\right], \tag{5.21}$$

$$\mathbb{E}_{\hat{\mathbf{Y}}\sim\mathcal{U}(\mathcal{Y}_j)}\left[\mu(\hat{\mathbf{Y}}, \mathbf{Y})\right] \leq \max_{j'\in\{0,\ldots,M\}} \mathbb{E}_{\hat{\mathbf{Y}}\sim\mathcal{U}(\mathcal{Y}_{j'})}\left[\mu(\hat{\mathbf{Y}}, \mathbf{Y})\right]. \tag{5.22}$$

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 Equations (5.21) and (5.22) that

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

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

Consequently, we have found a lower and upper bound for Equations (5.12) and (5.13), respectively. Namely,

$$\min_{h_M \in \mathcal{H}_M^{i.i.}} \left\{ \mathbb{E}_{\pi \sim \mathcal{U}(S_M)} \left[\mathbb{E}_{r \in \mathbb{R}} \left[\mu(h_M(\mathbf{X}_\pi, r), \mathbf{Y}_\pi) \right] \right] \right\} \geq \min_{j \in \{0,\dots,M\}} \left\{ \mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}(\mathcal{Y}_j)} \mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right\}, \quad (5.23)$$

$$\max_{h_M \in \mathcal{H}_M^{i.i.}} \left\{ \mathbb{E}_{\pi \sim \mathcal{U}(S_M)} \left[\mathbb{E}_{r \in \mathbb{R}} \left[\mu(h_M(\mathbf{X}_\pi, r), \mathbf{Y}_\pi) \right] \right] \right\} \leq \max_{j \in \{0,\dots,M\}} \left\{ \mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}(\mathcal{Y}_j)} \mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right\}. \quad (5.24)$$

Equality only holds for any classifier $h_M \in \mathcal{H}_M^{i.i.}$, when all probability mass is given to the set of minimizers and maximizers of $\left\{ \mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}(\mathcal{Y}_j)} \mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right\}$, respectively. In other words, the minimum can only be attained if

$$\sum_{j_{\min} \in \arg\min_{j \in \{0,\dots,M\}} \left\{ \mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}(\mathcal{Y}_j)} \mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right\}} \mathbb{P}(h_M(\cdot) \in \mathcal{Y}_{j_{\min}}) = 1, \quad (5.25)$$

and the maximum only if

$$\sum_{j_{\max} \in \arg\max_{j \in \{0,\dots,M\}} \left\{ \mathbb{E}_{\hat{\mathbf{Y}} \sim \mathcal{U}(\mathcal{Y}_j)} \mu(\hat{\mathbf{Y}}, \mathbf{Y}) \right\}} \mathbb{P}(h_M(\cdot) \in \mathcal{Y}_{j_{\max}}) = 1. \quad (5.26)$$

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

Remember that the Dutch Draw selects the optimal classifier based on Equations (5.10) and (5.11), which leads to

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

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

Combining this with the alternative definition of the Dutch Draw (Equation (5.9)) directly gives that

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

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

This shows in combination with Equations (5.25) and (5.26) that the optimal Dutch Draw classifier *actually* attains the bound given in Equations (5.23) and (5.24). It now follows that,

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

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

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

5.6 Discussion and conclusion

A baseline is crucial to assess the performance of a prediction model. However, there are infinitely many ways to devise a baseline method. As a necessary check in the development process, Van de Bijl et al. [14] 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. We have shown that, out of all *input-independent* binary classifiers, the Dutch Draw baseline is *permutation-optimal* for any *positional-invariant* measure. Our findings improve the evaluation process of any new binary classification method, as we have proven that the Dutch Draw 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 Dutch Draw baseline. First of all, we have considered 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 Dutch Draw 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 [14] that the Dutch Draw 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 less harmful instances and more normal instances [186]. There are thus many ways to estimate if $P \geq M/2$ holds. Nevertheless, even if the Dutch Draw baseline uses this information (only for the *accuracy*), we believe that any newly developed model should still beat the Dutch Draw 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 [65], probabilistic classification, or regression (between $[0, 1]$). Is a strategy similar to the Dutch Draw 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. Van de Bijl et al. [14] stated that the Dutch Draw baseline could be used to scale existing measures. This research provides more motivation to scale measures with the Dutch Draw baseline and not by using any other *input-independent* classifier. Yet, it could still be investigated how each measure should be scaled in order to maximize the explainability behind a performance score.

Part III

Quantifying the Relationships between Random Variables

Chapter 6

The Berkelmans-Pries Dependency Function: A Generic Measure of Dependence between Random Variables

Contents

Based on *Guus Berkelmans, Joris Pries, Sandjai Bhulai, and Rob van der Mei* (2023): "**The BP dependency function: A generic measure of dependence between random variables**". Published in the *Journal of Applied Probability*. [13]

Abstract

Measuring and quantifying dependencies between random variables (RV's) can give critical insights into a dataset. Typical questions are: 'Do underlying relationships exist?', 'Are some variables redundant?', and 'Is some target variable Y highly or weakly dependent on variable X?' Interestingly, despite the evident need for a general-purpose measure of dependency between RV's, common practice of data analysis is that most data analysts use the *Pearson correlation coefficient* to quantify dependence between RV's, while it is well-recognized that the correlation coefficient is essentially a measure for *linear* dependency only. Although many attempts have been made to define more generic dependency measures, there is yet no consensus on a standard, general-purpose dependency function. In fact, several ideal properties of a dependency function have been proposed, but without much argumentation. Motivated by this, in this chapter we will discuss and revise the list of desired properties and propose a new dependency function that meets all these requirements. This general-purpose dependency function provides data analysts a powerful means to quantify the level of dependence between variables. To this end, we also provide Python code to determine the dependency function for use in practice.

6.1 Introduction

In as early as 1958, Kruskal [93] stated that "There are infinitely many possible measures of association, and it sometimes seems that almost as many have been proposed at one time or another." Many years later, even more dependency measures have been suggested. Yet, and rather surprisingly, there still does not exist consensus on a general dependency function. Often the statement 'Y is dependent on X' means that Y is not independent of X. However, there are different levels of dependency. For example, random variable (RV) Y can be fully determined by RV X (i.e., $Y(\omega) = f(X(\omega))$ for all $\omega \in \Omega$ and for a measurable function f), or only partially.

But how should we quantify how much Y is dependent on X? Intuitively, and assuming that the dependency measure is normalized to the interval [0,1], one would say that if Y is fully determined by X then the dependency of Y w.r.t. X is as strong as possible, and so the dependency measure should be 1. On the other side of the spectrum, if X and Y are independent, then the dependency measure should be 0; and vice versa, it is desirable that dependence 0 *implies* that X and Y are stochastically independent. Note that the commonly used *Pearson correlation coefficient* does not meet these requirements. In fact, many examples exists where Y is fully determined by X while the correlation is zero.

Taking a step back, why is it actually useful to examine dependencies in a dataset? Measuring dependencies between the variables can lead to critical insights, which will lead to improved data analysis. First of all, it can reveal important explanatory relationships. How do certain variables interact? If catching a specific disease is highly dependent on the feature value of variable X, research should be done to investigate if this information can be exploited to reduce the number of patients with this disease. For example, if hospitalization time is dependent on a healthy lifestyle, measures can be taken to try to improve the overall fitness of a population. Dependencies can therefore function as an actionable steering rod. It is however important to keep in mind that dependency does not always mean causality. Dependency relations can also occur due to mere coincidence or as a byproduct of another process.

Dependencies can also be used for dimensionality reduction. If Y is highly dependent on X, not much information is lost when only X is used in the dataset. In this way, redundant variables or variables that provide little

additional information, can be removed to reduce the dimensionality of the dataset. With fewer dimensions, models can be trained more efficiently.

In these situations a dependency function can be very useful. However, finding the proper dependency function can be hard, as many attempts have already been made. In fact, most of us have a 'gut feeling' for what a dependency function should entail. To make this feeling more mathematically sound, Rényi [141] proposed a list of ideal properties for a dependency function. A long list of follow-up papers (see the references in Table 6.2.1 below) use this list as the basis for a wish list, making only minor changes to it, adding or removing some properties.

In view of the above, the contribution of this research is threefold:

- We determine a new list of ideal properties for a dependency function;
- We present a new dependency function and show that it fulfills all requirements;
- We provide Python code to determine the dependency function for the discrete and continuous case [132].

The remainder of this chapter is organized as follows. In Section 6.2, we summarize which ideal properties have been stated in previous literature. By critically assessing these properties, we derive a new list of ideal properties for a dependency function (see Table 6.2.2), which lays the foundation for a new search for a general-purpose dependency function. In Section 6.3, the properties are checked for existing methods, and we conclude that there does not yet exist a dependency function that has all desired properties. Faced by this, in Section 6.4 we define a new dependency function and show in Section 6.5 that this function meets all the desired properties. Finally, Section 6.6 outlines the general findings and addresses possible future research opportunities.

6.2 Desired properties of a dependency function

What properties should an ideal dependency function have? In this section, we summarize previously suggested properties. Often, these characteristics are posed without much argumentation. Therefore, we analyze and discuss which properties are actually ideal and which properties are to be believed not relevant, or even wrong.

In Table 6.2.1, a summary is given of (twenty-two) 'ideal properties' found in previous literature, grouped into five different categories. We denote these by I.1-22. From these properties we derive a new set of desirable properties

denoted by II.1-8, see Table 6.2.2. Next, we discuss the properties suggested in previous literature and how the new list is derived from them.

Desired property II.1 (Asymmetry):
At first glance, it seems obvious that a dependency function should adhere to property I.13 and be symmetric. However, this is a common misconception for the dependency function. Y can be fully dependent on X, but this does not mean that X is fully dependent on Y. Lancaster [100] indirectly touched upon this same point by defining *mutual complete dependence*. First it is stated that Y is *completely dependent* on X if $Y = f(X)$. X and Y are called *mutually completely dependent* if X is completely dependent on Y and vice versa. Thus, this indirectly shows that dependence should not necessarily be symmetric, otherwise the extra definition would be redundant. In [100] the following great asymmetric example was given.

Example 6.2.1. Let $X \sim \mathcal{U}(0,1)$ be uniformly distributed and let $Y = -1$ if $X \leq \frac{1}{2}$ and $Y = 1$ if $X > \frac{1}{2}$.

Then, Y is fully dependent on X, but not vice versa. To drive this point home even more, we give another asymmetric example.

Example 6.2.2. X is uniformly randomly drawn out of $\{1,2,3,4\}$ and $Y := X \mod 2$.

Y is fully dependent on X, because given X the value of Y is deterministically known. On the other hand, X is not completely known given Y. Note that $Y = 1$ still leaves the possibility for $X = 1$ or $X = 3$. Thus, when assessing the dependency between variable X and variable Y, Y is fully dependent on X, whereas X is not fully dependent on Y. In other words, $\mathrm{Dep}(X,Y) \neq \mathrm{Dep}(Y,X)$.

In conclusion, *an ideal dependency function should not always be symmetric.* To emphasize this point even further, we change the notation of the dependency function. Instead of $\mathrm{Dep}(X,Y)$, we will denote $\mathrm{Dep}(Y|X)$ for how much Y is dependent on X. Based by this, property I.13 is changed into II.1.

Desired property II.2 (Range):
An ideal dependency function should be scaled to the interval $[0,1]$. Otherwise, it can be very hard to draw meaningful conclusions from a dependency score without a known maximum or minimum. What would a score of 4.23 mean without any information about the possible range? Therefore, property I.1 is retained. A special note on the range for the well-known *Pearson correlation coefficient* [131], which is $[-1,1]$: The negative or posi-

tive sign denotes the direction of the linear correlation. When examining more complex relationships, it is unclear what 'direction' entails. We believe that a dependency function should measure by *how much* variable Y is dependent on X, and not necessarily in which way. In summary, we require: $0 \leq \text{Dep}(Y|X) \leq 1$.

Desired property II.3 (Independence and dependency 0):
If Y is independent of X, it should hold that the dependency achieves the lowest possible value, namely zero. Otherwise, it is vague what a dependency score lower than the dependency between two independent variables means. A major issue of the commonly used *Pearson correlation coefficient*, is that zero correlation does not imply independence. This makes it complicated to derive conclusions from a correlation score. Furthermore, note that if Y is independent of X, it should automatically hold that X is also independent of Y. In this case, X and Y are independent, because otherwise some dependency relation should exist. Thus, we require: $\text{Dep}(Y|X) = 0 \iff X$ and Y are independent.

Desired property II.4 (Functional dependence and dependency 1):
If Y is strictly dependent on X (and thus fully determined by X), the highest possible value should be attained. It is otherwise unclear what a higher dependency would mean. However, it is too restrictive to demand that the dependency is only 1 if Y is strictly dependent on X. Rényi [141] stated "It seems at the first sight natural to postulate that $\delta(\xi, \eta) = 1$ only if there is a strict dependence of the mentioned type between ξ and η, but this condition is rather restrictive, and it is better to leave it out". Take, for example, $Y \sim \mathcal{U}(-1, 1)$ and $X := Y^2$. Knowing X reduces the infinite set of possible values for Y to only two $\left(\pm\sqrt{X}\right)$, whereas it would reduce to one if Y was fully determined by X. It would be really restrictive to enforce $\text{Dep}(Y|X) < 1$, as there is only an infinitesimal difference compared to the strictly dependent case. Summarizing, we require: $Y = f(X) \rightarrow \text{Dep}(Y|X) = 1$.

Desired property II.5 (Unambiguity):
Kruskal [93] stated "It is important to recognize that the question 'Which single measure of association should I use?,' is often unimportant. There may be no reason why two or more measures should not be used; the point I stress is that, whichever ones are used, they should have clear-cut population interpretations." It is very important that a dependency score leaves no room for ambiguity. The results should stroke with our natural expectation. Therefore, we introduce a new requirement based on a simple example: suppose we have a number of independent RV's and observe one of

these at random. The dependency of each random variable on the observed variable should be equal to the probability it is picked. More formally, let $Y_1, Y_2, \ldots, Y_N, S$ be independent variables with S a selection variable s.t. $\mathbb{P}(S = i) = p_i$ and $\sum_{i=1}^{N} p_i = 1$. When X is defined as $X = \sum_{i=1}^{N} \mathbb{1}_{S=i} \cdot Y_i$, it should hold that $\mathrm{Dep}(Y_i|X) = p_i$ for all $i \in \{1, \ldots, N\}$. Simply said, the dependency function should give desired results in specific situations, where we can argue what the outcome should be. This is one of these cases.

Desired property II.6 (Generally applicable):
Our aim is to find a general dependency function, which we denote by $Dep(X|Y)$. This function must be able to handle all kinds of variables: *continuous*, *discrete*, and *categorical* (even nominal). These types of variables occur frequently in a dataset. A general dependency function should be able to measure the dependency of a categorical variable Y on a continuous variable X. Stricter than I.9-12, we want a single dependency function that is applicable to any combination of these variables.

There is one exception to this generality. In the case that Y is almost surely constant it is completely independent as well as completely determined by X. Arguing what the value of a dependency function should be in this case is a bit similar to arguing the value of $\frac{0}{0}$. Therefore, we argue that in this case it should be either undefined or return some value that represents the fact that Y is almost surely constant (for example -1 since this cannot be normally attained).

Desired property II.7 (Invariance under isomorphisms):
Properties I.14-20 discuss when the dependency function should be invariant. Most are only meant for variables with an ordering, as 'strictly increasing', 'translation' and 'scaling' are otherwise ill-defined. As the dependency function should be able to handle nominal variables, we assume that the dependency is invariant under isomorphisms, see II.7. Note that this is a stronger assumption than I.14-20. Compare Example 6.2.2 with the following example.

Example 6.2.3. Let X' be uniformly randomly drawn out of $\{\circ, \triangle, \square, \Diamond\}$ and $Y' = \clubsuit$ if $X' \in \{\circ, \square\}$ and $Y' = \spadesuit$ if $X' \in \{\triangle, \Diamond\}$.

It should hold that $\mathrm{Dep}(Y|X) = \mathrm{Dep}(Y'|X')$ and $\mathrm{Dep}(X|Y) = \mathrm{Dep}(X'|Y')$, as the relationship between the variables is the same (only altered using isomorphisms). So, for any isomorphisms f and g we require $\mathrm{Dep}(g(Y)|f(X)) = \mathrm{Dep}(Y|X)$.

Desired property II.8 (Non-increasing under functions of X): Additionally, $\mathrm{Dep}(Y|X)$ should not increase if a measurable function f is applied to X since any dependence on $f(X)$ corresponds to a dependence on X (but not necessarily the other way around). The information gained from knowing X can only be reduced, never increased by applying a function.

However, though it might be natural to expect the same for functions applied to Y, consider once again Example 6.2.2 (but with X and Y switched around) and the following 2 functions: $f_1(Y) := Y \mod 2$ and $f_2(Y) := \lceil \frac{Y}{2} \rceil$. Then $f_1(Y)$ is completely predicted by X and should therefore have a dependency of 1 while $f_2(Y)$ is independent of X and should therefore have a dependency of 0. So the dependency should be free to increase or decrease for functions applied to Y. To conclude, for any measurable function f we require: $\mathrm{Dep}(Y|f(X)) \leq \mathrm{Dep}(Y|X)$.

Exclusion of Pearson correlation coefficient as a special case: According to properties I.21-22, when X and Y are normally distributed the dependency function should coincide with or be a function of the *Pearson correlation coefficient*. However, these properties lack a good argumentation for why this would be ideal. It is not obvious why this would be a necessary condition. Even more, there are many known problems and pitfalls with the correlation coefficient [48, 78], so it seems undesirable to force an ideal dependency function to reduce to a function of the correlation coefficient, when the variables are normally distributed. This is why we leave these properties out.

6.3 Assessment of the desired properties for existing dependency measures

In this section, we assess whether existing dependency functions have the properties listed above. In doing so, we limit this section to the most commonly used dependency measures. Table 6.3.1 shows which properties each investigated measure adheres to.

Although the desired properties listed in Table 6.2.2 seem not too restrictive, many dependency measures fail to have many of these properties. One of the most commonly used dependency measures, the *Pearson correlation*

Table 6.2.1: Desirable properties literature: A summary of desirable properties for a dependency function stated in previous literature.

Property group	Property	Article(s)
Range	I.1. $0 \leq \text{Dep}(X,Y) \leq 1$	[4, 48, 66, 68, 79, 141, 142, 168, 172]
	I.2. $\text{Dep}(X,Y) = 0 \Leftarrow X$ and Y are independent	[66, 79, 142]
	I.3. $\text{Dep}(X,Y) = 0 \Rightarrow X$ and Y are independent	[172]
	I.4. $\text{Dep}(X,Y) = 0 \Leftrightarrow X$ and Y are independent	[4, 48, 68, 121, 141, 168]
	I.5. $\text{Dep}(X,Y) = 1 \Leftrightarrow Y = LX$ with probability 1, where L is a similarity transformation	[121]
	I.6. $\text{Dep}(X,Y) = 1 \Leftarrow X$ and Y are strictly dependent	[4, 66, 141, 142]
	I.7. $\text{Dep}(X,Y) = 1 \Leftrightarrow X$ and Y are comonotonic or countermonotonic	[48]
	I.8. $\text{Dep}(X,Y) = 1 \Leftrightarrow X$ and Y are strictly dependent	[68]
General	I.9. $\text{Dep}(X,Y)$ is defined for any X,Y where both are not constant	[68, 121, 141]
	I.10. Well-defined for both continuous and discrete variables	[66]
	I.11. Defined for both categorical and continuous variables; and for ordinal categorical variables for which there may be underlying continuous variables	[79]
	I.12. There is a close relationship between the measure for the continuous variables and the measure for the discretization of the variables	[79]
Symmetric	I.13. $\text{Dep}(X,Y) = \text{Dep}(Y,X)$	[4, 48, 141, 142, 168]
Applying function to argument	I.14. $\text{Dep}(f(X),g(Y)) = \text{Dep}(X,Y)$ with f,g strictly monotonic functions	[4]
	I.15. $\text{Dep}(f(X),Y) = \text{Dep}(X,Y)$ with $f : \mathbb{R} \to \mathbb{R}$ strictly monotonic on the range of X	[48]
	I.16. $\text{Dep}(f(X),f(Y)) = \text{Dep}(X,Y)$ with f continuous and strictly increasing	[66, 168]
	I.17. $\text{Dep}(f(X),g(Y)) = \text{Dep}(X,Y)$ if $f(\cdot),g(\cdot)$ map the real axis in a one-to-one way onto itself	[79, 141]
	I.18. $\text{Dep}(X,Y)$ is invariant with respect to all similarity transformations	[121]
	I.19. $\text{Dep}(X,Y)$ is invariant with respect to translation and scaling	[168]
	I.20. $\text{Dep}(X,Y)$ is scale invariant	[172]
Behavior normal distribution	I.21. $\text{Dep}(X,Y)$ is a function of the Pearson correlation if the joint distribution of X and Y is normal	[4, 66, 172]
	I.22. $\text{Dep}(X,Y) = \lvert\rho(X,Y)\rvert$ if the joint distribution of X and Y is normal, where ρ is the Pearson correlation	[79, 141]

Table 6.2.2: Revised list of desirable properties: New list of desirable properties for a dependency function.

Property group	Property
Asymmetric	II.1. There exist RV's X, Y such that $\mathrm{Dep}(Y\|X) \neq \mathrm{Dep}(X\|Y)$.
Intuitive	II.2. $0 \leq \mathrm{Dep}(Y\|X) \leq 1$ for all RV's X and Y.
	II.3. $\mathrm{Dep}(Y\|X) = 0 \Leftrightarrow X$ and Y are independent.
	II.4. $\mathrm{Dep}(Y\|X) = 1 \Leftarrow Y$ is strictly dependent on X.
	II.5. If $Y_1, Y_2, \ldots, Y_N, S$ independent with $\mathbb{P}(S \in [N]) = 1$, $\mathbb{P}(S = i) = p_i$ and $X = Y_S$ then $\mathrm{Dep}(Y_i\|X) = p_i$ must hold.
General	II.6. Applicable for any combination of continuous, discrete and categorical RV's X, Y, where Y is not a.s. constant.
Functions	II.7. $\mathrm{Dep}(g(Y)\|f(X)) = \mathrm{Dep}(Y\|X)$ for any isomorphisms f, g.
	II.8. $\mathrm{Dep}(Y\|f(X)) \leq \mathrm{Dep}(Y\|X)$ for any measurable function f.

coefficient, does not even satisfy any one of the desirable properties. Furthermore, almost all measures are not asymmetric. The one measure that comes closes to fulfilling all requirements, is the *uncertainty coefficient* [131]. This is a normalized asymmetric variant of the *mutual information* [131], where the discrete variant is defined as

$$C_{XY} = \frac{I(X,Y)}{H(Y)} = \frac{\sum_{x,y} p_{X,Y}(x,y) \log\left(\frac{p_{X,Y}(x,y)}{p_X(x)\cdot p_Y(y)}\right)}{-\sum_y p_Y(y)\log(p_Y(y))},$$

where $H(Y)$ is the entropy of Y and $I(X,Y)$ is the mutual information of X and Y. Note that we use the following notation $p_X(x) = \mathbb{P}(X = x)$, $p_Y(y) = \mathbb{P}(Y = y)$, and $p_{X,Y}(x,y) = \mathbb{P}(X = x, Y = y)$ throughout the chapter. In addition, for a set H we define $p_X(H) = \mathbb{P}(X \in H)$ (and similarly for p_Y and $p_{X,Y}$).

However, the *uncertainty coefficient* does not satisfy properties II.5 and II.6. For example, if $Y \sim \mathcal{U}(0,1)$ is uniformly drawn, the entropy of Y

becomes:

$$\begin{aligned} H(Y) &= -\int_0^1 f_Y(y) \ln\left(f_Y(y)\right) dy \\ &= -\int_0^1 1 \cdot \ln(1)\, dy \\ &= 0. \end{aligned}$$

Thus, for any X, the uncertainty coefficient is now undefined (division by zero). Therefore, the uncertainty coefficient is not as generally applicable as property II.6 requires.

Two other measures that satisfy many (but not all) properties are *mutual dependence* [4] and *maximal correlation* [58]. Mutual dependence is defined as the Hellinger distance [69] d_h between the joint distribution and the product of the marginal distributions, defined as follows (cf. [4]):

$$d(X,Y) \triangleq d_h(f_{XY}(x,y), f_X(x) \cdot f_Y(y)). \tag{6.1}$$

Maximal correlation is defined as (cf. [141]):

$$S(X,Y) = \sup_{f,g} R(f(X), g(Y)), \tag{6.2}$$

where R is the Pearson correlation coefficient, and where f, g are Borel measurable functions, such that $R(f(X), g(Y))$ is defined [141].

Clearly, Equations (6.1) and (6.2) are symmetric. The joint distribution and the product of the marginal distributions does not change by switching X and Y. Furthermore, the Pearson correlation coefficient is symmetric, making the maximal correlation also symmetric. Therefore, both measures do not have property II.1.

There are two more measures (one of which is a variation of the other) which satisfy many (but not all) properties, and additionally closely resemble the measure we intend to propose. Namely, the *strong mixing coefficient* [20]

$$\alpha(X,Y) = \sup_{A \in \mathcal{E}_X, B \in \mathcal{E}_Y} \left\{ \left| \mu_{X,Y}(A \times B) - \mu_X(A)\mu_Y(B) \right| \right\},$$

and it's relaxation, the *β-mixing coefficient* [20]

$$\beta(X,Y) = \sup\left\{\frac{1}{2}\sum_{i=1}^{I}\sum_{j=1}^{J}|(\mu_{X,Y}(A_i \times B_j) - \mu_X(A_i)\mu_Y(B_j))|\right\},$$

where the supremum is taken over all finite partitions $(A_1, A_2, \ldots, A_I)$ and $(B_1, B_2, \ldots, B_J)$ of E_X and E_Y with $A_i \in \mathcal{E}_X$ and $B_j \in \mathcal{E}_Y$. However, these measures fail the properties II.1, II.4, and II.5.

Table 6.3.1: Assessment of properties: Properties of previous dependencies functions (✓= satisfied, ✗= *not* satisfied). See [12] for the proofs.

Measure	Asymmetric	Intuitive				General	Functions	
	II.1	II.2	II.3	II.4	II.5	II.6	II.7	II.8
Pearson correlation coefficient [131]	✗	✗	✗	✗	✗	✗	✗	✗
Spearman's rank correlation coefficient [131]	✗	✗	✗	✗	✗	✗	✗	✗
Kendall rank correlation coefficient [131]	✗	✗	✗	✗	✗	✗	✗	✗
Mutual information [131]	✗	✗	✓	✗	✗	✗	✓	✓
Uncertainty coefficient [131]	✓	✓	✓	✓	✗	✗	✓	✓
Total correlation [183]	✗	✗	✓	✗	✗	✗	✓	✓
Mutual dependence [4]	✗	✓	✓	✗	✗	✓	✓	✓
Δ_{L_1} [29]	✗	✗	✓	✗	✗	✓	✓	✓
Δ_{SD} [29]	✗	✗	✓	✗	✗	✗	✗	✗
Δ_{ST} [29]	✗	✗	✗	✗	✗	✗	✗	✗
Monotone correlation [87]	✗	✓	✓	✗	✗	✗	✗	✗
Maximal correlation [58]	✗	✓	✓	✓	✗	✓	✓	✓
Distance correlation [172]	✗	✓	✓	✗	✗	✗	✗	✗
Maximum canonical correlation (first) [72]	✗	✓	✗	✗	✗	✗	✗	✗
Strong mixing coefficient [20]	✗	✓	✓	✗	✗	✓	✓	✓
β-mixing coefficient [20]	✗	✓	✓	✗	✗	✓	✓	✓

6.4 The Berkelmans-Pries dependency function

After devising a new list of ideal properties (see Table 6.2.2) and showing that these properties are not fulfilled by existing dependency functions (see Table 6.3.1), we will now introduce a new dependency function that will meet all requirements. Throughout, we will refer to this function as the *Berkelmans-Pries (BP)* dependency function.

The key question surely is: What is dependency? Although this question deserves an elaborate philosophical study, we believe that measuring the dependency of Y on X, is essentially measuring how much the distribution of Y changes on average based on the knowledge of X, divided by the maximum possible change. This is the key insight, where the *BP* dependency function is based on. To measure this, we first have to determine the difference between the distribution of Y *with* and *without* conditioning on the value of X times the probability that X takes on this value in Section 6.4.1. Secondly, we have to measure what the maximum possible change in probability mass is, which is used to properly scale the dependency function and make it asymmetric (see Section 6.4.2).

6.4.1 Definition expected absolute change in distribution

We start by measuring the *expected absolute change in distribution* (UD), which is the difference between the distribution of Y *with* and *without* conditioning on the value of X times the probability that X takes on this value. For discrete RV's, we obtain the following definition.

Definition 6.4.1 (Discrete UD)**.** For any discrete RV's X and Y,

$$\mathrm{UD}\left(X,Y\right) := \sum_{x} p_X(x) \cdot \sum_{y} \left| p_{Y|X=x}(y) - p_Y(y) \right|.$$

More explicit formulations of UD for specific combinations of RV's are given in Section 6.A. For example, when X and Y remain discrete and take values in E_X and E_Y, respectively, it can equivalently be defined as:

$$\mathrm{UD}\left(X,Y\right) := 2 \sup_{A \subset E_X \times E_Y} \left\{ \sum_{(x,y)\in A} \left(p_{X,Y}(x,y) - p_X(x) \cdot p_Y(y) \right) \right\}.$$

Similarly, for continuous RV's, we obtain the following definition for UD.

Definition 6.4.2 (Continuous UD)**.** For any continuous RV's X and Y,

$$\mathrm{UD}\left(X,Y\right) := \int_{\mathbb{R}} \int_{\mathbb{R}} |f_{X,Y}(x,y) - f_X(x) f_Y(y)| dy dx.$$

Note that this is the same as Δ_{L_1} [29].

In the general case, UD is defined in the following manner.

Definition 6.4.3 (General UD)**.** For any $X : (\Omega, \mathcal{F}, \mu) \to (E_X, \mathcal{E}(X))$ and $Y : (\Omega, \mathcal{F}, \mu) \to (E_Y, \mathcal{E}(Y))$, UD is defined as

$$\mathrm{UD}\,(X, Y) := 2 \sup_{A \in \mathcal{E}(X) \bigotimes \mathcal{E}(Y)} \left\{ \mu_{(X,Y)}(A) - (\mu_X \times \mu_Y)(A) \right\},$$

where $\mathcal{E}(X) \bigotimes \mathcal{E}(Y)$ is the σ-algebra generated by the sets $C \times D$ with $C \in \mathcal{E}(X)$ and $D \in \mathcal{E}(Y)$. Furthermore, $\mu_{(X,Y)}$ denotes the joint probability measure on $\mathcal{E}(X) \bigotimes \mathcal{E}(Y)$ and $\mu_X \times \mu_Y$ is the product measure.

6.4.2 Maximum UD given Y

Next, we have to determine the maximum of UD for a fixed Y in order to scale the dependency function to $[0, 1]$. To this end, we prove that for a given Y:

$$X \text{ fully determines } Y \Rightarrow \mathrm{UD}\,(X, Y) \geq \mathrm{UD}\left(X', Y\right),$$

for any RV X'.

The full proof for the general case is given in Section 6.B.4, which uses the upper bound determined in Section 6.B.3. However, we will show the discrete case here to give some intuition about the proof. We define the following helpful set $C_y := \{x | p_{X,Y}(x, y) \geq p_X(x) \cdot p_Y(y)\}$, which leads to

$$\begin{aligned}
\mathrm{UD}\,(X, Y) &= 2 \sum_y \left(p_{X,Y}(C_y \times \{y\}) - p_X(C_y) \cdot p_Y(y) \right) \\
&\leq 2 \sum_y \left(\min\left\{ p_X(C_y), p_Y(y) \right\} - p_X(C_y) \cdot p_Y(y) \right) \\
&= 2 \sum_y \left(\min\left\{ p_X(C_y) \cdot (1 - p_Y(y)), (1 - p_X(C_y)) \cdot p_Y(y) \right\} \right) \\
&\leq 2 \sum_y \left(p_Y(y) \cdot (1 - p_Y(y)) \right) \\
&= 2 \sum_y \left(p_Y(y) - p_Y(y)^2 \right) \\
&= 2 \cdot (1 - \sum_y p_Y(y)^2),
\end{aligned}$$

with equality iff both inequalities are equalities. Which occurs iff for all y it holds that $p_{X,Y}(C_y \times \{y\}) = p_X(C_y) = p_Y(y)$. So we have equality when for all y the set C_y has the property that $x \in C_y$ iff $Y = y$. Or equivalently $Y = f(X)$ for some function f. Thus,

$$UD(X,Y) \leq 2 \cdot (1 - \sum_y p_Y(y)^2),$$

with equality iff $Y = f(X)$ for some function f.

Note that this holds for every X that fully determines Y. In particular, for $X := Y$ it now follows that

$$\mathrm{UD}(Y,Y) = 2 \cdot (1 - \sum_y p_Y(y)^2) \geq \mathrm{UD}\left(X', Y\right),$$

for any RV X'.

6.4.3 Definition Berkelmans-Pries dependency function

Finally, we can define the *BP* dependency function to measure how much Y is dependent on X. We call a random variable Y *non-trivial* if E_Y is *not* an atom and *trivial* when E_Y is an atom (in most practical cases this is the same as a random variable being a.s. constant).

Definition 6.4.4 (BP dependency function)**.** For any RV's X and Y the *Berkelmans-Pries dependency function* is defined as

$$\mathrm{Dep}(Y|X) := \begin{cases} \frac{\mathrm{UD}(X,Y)}{\mathrm{UD}(Y,Y)} & \text{if } Y \text{ is non-trivial,} \\ \text{undefined} & \text{if } Y \text{ is trivial.} \end{cases}$$

This is the difference between the distribution of Y *with* and *without* conditioning on the value of X times the probability that X takes on this value divided by the largest possible difference for an arbitrary X'. Note that $\mathrm{UD}(Y,Y) = 0$ if and only if Y is almost surely constant (see Section 6.B.4), which leads to division by zero. However, we previously argued in Section 6.2 that if Y is almost surely constant, it is completely independent as well as completely determined by X. It should therefore be undefined.

6.5 Properties of the Berkelmans-Pries dependency function

Next, we show that our new *BP* dependency function satisfies all requirements from Table 6.2.2. To this end, we use properties of UD (see Section 6.B) to derive properties II.1-8.

Property II.1 (Asymmetry): Observe that it holds for Example 6.2.1 that $\mathrm{UD}(X,Y) = 1$, $\mathrm{UD}(X,X) = 2$, and $\mathrm{UD}(Y,Y) = 1$. Thus,

$$\mathrm{Dep}(Y|X) = \frac{\mathrm{UD}(X,Y)}{\mathrm{UD}(Y,Y)} = 1,$$

$$\mathrm{Dep}(X|Y) = \frac{\mathrm{UD}(X,Y)}{\mathrm{UD}(X,X)} = \frac{1}{2}.$$

Therefore, we see that $\mathrm{Dep}(Y|X) \neq \mathrm{Dep}(X|Y)$ for this example, thus making the *BP* dependency asymmetric.

Property II.2 (Range): In Section 6.B.2, we show that for every X, Y it holds that $\mathrm{UD}(X,Y) \geq 0$. Furthermore, in Section 6.B.3 we prove that $\mathrm{UD}(X,Y) \leq 2\left(1 - \sum_{y \in d_Y} \mu_Y(\{y\})^2\right)$ for all RV's X. In Section 6.B.4 we show for almost all cases that this bound is tight for $\mathrm{UD}(Y,Y)$. Thus, it must hold that $0 \leq \mathrm{UD}(X,Y) \leq \mathrm{UD}(Y,Y)$ and it then immediately follows that $0 \leq \mathrm{Dep}(Y|X) \leq 1$.

Property II.3 (Independence and dependency 0): In Section 6.B.2, we prove that

$$\mathrm{UD}(X,Y) = 0 \Leftrightarrow X \text{ and } Y \text{ are independent.}$$

Furthermore, observe that $\mathrm{Dep}(Y|X) = 0$ if and only if $\mathrm{UD}(X,Y) = 0$. Thus,

$$\mathrm{Dep}(Y|X) = 0 \Leftrightarrow X \text{ and } Y \text{ are independent.}$$

Property II.4 (Functional dependence and dependency 1): In Section 6.B.4, we show that if X fully determines Y and X' is any RV we have that $\mathrm{UD}(X,Y) \geq \mathrm{UD}(X',Y)$. This holds in particular for $X := Y$. Thus, if X fully determines Y it follows that $\mathrm{UD}(X,Y) = \mathrm{UD}(Y,Y)$, so

$$\mathrm{Dep}(Y|X) = \frac{\mathrm{UD}(X,Y)}{\mathrm{UD}(Y,Y)} = 1.$$

In conclusion: if there exists a measurable function f such that $Y = f(X)$, then $\mathrm{Dep}\,(Y|X) = 1$

Property II.5 (Unambiguity): We show the result for discrete RV's below. For the proof of the general case see Section 6.B.5. Let E be the range of the independent $Y_1, Y_2, \ldots, Y_N$. By definition, it holds that $\mathbb{P}(X = x) = \sum_j \mathbb{P}(Y_j = x) \cdot \mathbb{P}(S = j)$. Thus, for all $i \in \{1, \ldots N\}$ we have

$$\begin{aligned}
\mathrm{UD}\,(X, Y_i) &= 2 \sup_{A \subset E \times E} \left\{ \sum_{(x,y) \in A} \left(\mathbb{P}(X = x, Y_i = y) - \mathbb{P}(X = x)\mathbb{P}(Y_i = y) \right) \right\} \\
&= 2 \sup_{A \subset E \times E} \left\{ \sum_{(x,y) \in A} \Big(\sum_j \mathbb{P}(Y_j = x, Y_i = y, S = j) \right. \\
&\qquad \left. - \mathbb{P}(X = x)\mathbb{P}(Y_i = y) \Big) \right\} \\
&= 2 \sup_{A \subset E \times E} \left\{ \sum_{(x,y) \in A} \Big(\sum_{j \neq i} \mathbb{P}(Y_j = x)\mathbb{P}(Y_i = y)\mathbb{P}(S = j) \right. \\
&\qquad + \mathbb{P}(Y_i = x, Y_i = y)\mathbb{P}(S = i) \\
&\qquad \left. - \sum_j \mathbb{P}(Y_j = x)\mathbb{P}(S = j)\mathbb{P}(Y_i = y) \Big) \right\} \\
&= 2 \sup_{A \subset E \times E} \left\{ \sum_{(x,y) \in A} \Big(p_i \mathbb{P}(Y_i = x, Y_i = y) \right. \\
&\qquad \left. - p_i \mathbb{P}(Y_i = x)\mathbb{P}(Y_i = y) \Big) \right\} \\
&= p_i \cdot \mathrm{UD}\,(Y_i, Y_i)\,.
\end{aligned}$$

This leads to

$$\mathrm{Dep}\left(Y_i|X\right) = \frac{\mathrm{UD}\left(X, Y_i\right)}{\mathrm{UD}\left(Y_i, Y_i\right)} = \frac{p_i \cdot \mathrm{UD}\left(Y_i, Y_i\right)}{\mathrm{UD}\left(Y_i, Y_i\right)} = p_i.$$

Therefore, we can conclude that property II.5 holds.

Property II.6 (Generally applicable): The *BP* dependency measure can be applied for any combination of continuous, discrete and categorical variables. It can handle arbitrary many RV's as input by combining them. Thus, the *BP* dependency function is generally applicable.

Property II.7 (Invariance under isomorphisms): In Section 6.B.6, we prove that applying a measurable function to X or Y does not increase UD. Thus, it must hold for all isomorphisms f, g that

$$\begin{aligned}\mathrm{UD}\left(X, Y\right) &= \mathrm{UD}\left(f^{-1}(f(X)), g^{-1}(g(Y))\right)\\ &\leq \mathrm{UD}\left(f(X), g(Y)\right)\\ &\leq \mathrm{UD}\left(X, Y\right).\end{aligned}$$

Therefore, all inequalities are actually equalities. In other words,

$$\mathrm{UD}\left(f(X), g(Y)\right) = \mathrm{UD}\left(X, Y\right).$$

6

It now immediately follows for the *BP* dependency measure that

$$\begin{aligned}\mathrm{Dep}\left(g(Y)|f(X)\right) &= \frac{\mathrm{UD}\left(f(X), g(Y)\right)}{\mathrm{UD}\left(g(Y), g(Y)\right)}\\ &= \frac{\mathrm{UD}\left(X, Y\right)}{\mathrm{UD}\left(Y, Y\right)}\\ &= \mathrm{Dep}\left(Y|X\right),\end{aligned}$$

thus Property II.7 is satisfied.

Desired property II.8 (Non-increasing under functions of X): In Section 6.B.6, we prove that transforming X or Y using a measurable function does not increase UD. In other words, for any measurable function f, it holds that

$$\mathrm{UD}\left(f(X), Y\right) \leq \mathrm{UD}\left(X, Y\right).$$

Consequently, Property II.8 holds for the *BP* dependency function, as

$$\begin{aligned}\mathrm{Dep}\left(Y|f(X)\right) &= \frac{\mathrm{UD}\left(f(X),Y\right)}{\mathrm{UD}\left(Y,Y\right)} \\ &\leq \frac{\mathrm{UD}\left(X,Y\right)}{\mathrm{UD}\left(Y,Y\right)} \\ &= \mathrm{Dep}\left(Y|X\right).\end{aligned}$$

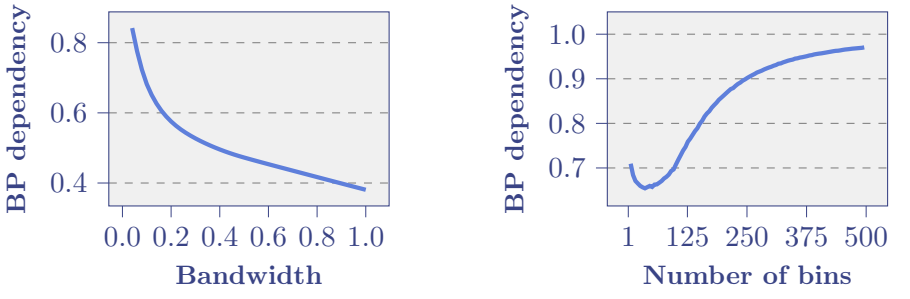
6.6 Discussion and conclusion

Motivated by the need to measure and quantify the level dependence between random variables, we have proposed a general-purpose dependency function. The function meets an extensive list of important and desired properties, and can be viewed as a powerful alternative to the classical Pearson correlation coefficient, which is often used by data analysts today.

Whilst it is recommended to use our new dependency function, it is important to understand the limitations and potential pitfalls of the new dependency function. Below we elaborate on these aspects.

The underlying probability density function of a RV is often unknown in practice; instead, a set of outcomes is observed. These samples can then be used (in a simple manner) to approximate any discrete distribution. However, this is generally not the case for continuous variables. There are mainly two categories for dealing with continuous variables: either (1) the observed samples are combined using kernel functions into a continuous function (*kernel density estimation* [64]), or (2) the continuous variable is reduced to a discrete variable using *data binning*. The new dependency measure can be applied thereafter.

A main issue is that the dependency measure is dependent of parameter choices of either *kernel density estimation* or *data binning*. To illustrate this, we conduct the following experiment: Let $X \sim \mathcal{U}(0,1)$ and define $Y = X + \epsilon$ with $\epsilon \sim \mathcal{N}(0, 0.1)$. Next, we draw 5,000 samples of X and ϵ and determine each corresponding Y. For *kernel density estimation*, we use Gaussian kernels with constant bandwidth. The result of varying the bandwidth on the dependency score can be seen in Figure 6.6.1a. With *data binning*, both X and Y are binned using bins with fixed size. Increasing or decreasing the number of bins changes the size of the bins. The impact



(a) Kernel density estimation **(b) Data binning**

Figure 6.6.1: Problematic parameter selection: Influence of chosen bandwidth (a) / number of bins (b) on the dependency score $\mathrm{Dep}(Y|X)$ with 5,000 samples of $X \sim \mathcal{U}(0,1)$ and $Y = X + \epsilon$ with $\epsilon \sim \mathcal{N}(0, 0.1)$.

of changing the number of bins on the dependency score, can be seen in Figure 6.6.1b.

The main observation from Figures 6.6.1a and 6.6.1b is that the selection of the parameters is important. In the case of the *kernel density estimation*, we see the traditional *trade-off* between *over-fitting* when the bandwidth is too small and *under-fitting* when the bandwidth is too large. On the other hand, with *data binning*, we see different behavior: Having too few bins seems to overestimate the dependency score and as bins increase the estimator of the dependency score decreases up to a certain point, where-after it starts increasing again. The bottom of the curve seems to be marginally higher than the true dependency score of 0.621.

This observation raises a range of interesting questions for future research. For example, are the dependency scores estimated by binning consistently higher than the true dependency? Is there a correction that can be applied to get an unbiased estimator? Is the minimum of this curve an asymptotically consistent estimator? Which binning algorithms give the closest approximation of the true dependency?

An interesting observation, with respect to kernel density estimation, is that it appears that at a bandwidth of 0.1 the estimator of the dependency score is close to the true dependency score of approximately 0.621. However, this parameter choice could only be made if the underlying probability process was known *a priori*.

Yet, there is another challenge with kernel density estimation, when X consists of many variables or feature values. Each time Y is conditioned

on a different value of X, either the density needs to be estimated again or the estimation of the joint distribution needs to be integrated. Both can rapidly become very time-consuming. When using data binning, it suffices to bin the data once. Furthermore, no integration is required making it much faster. Therefore, our current recommendation would be to bin the data and not use kernel density estimation.

Another exciting research avenue would be to fundamentally explore the set of functions that satisfy all desired dependency properties. Is the BP dependency the only measure that fulfills all conditions? If there exist two solutions, can we derive a new solution by smartly combining them? Without property II.5 any order-preserving bijection of $[0, 1]$ with itself would preserve all properties when applied to a solution. However, property II.5 does restrict the solution space. It remains an open problem if this is restrictive enough to result in a unique solution: the BP dependency.

Appendix (Chapter 6)

Notation

The following general notation is used throughout this appendix. Let X and Y be RV's, such that $X : (\Omega, \mathcal{F}, \mathbb{P}) \to (E_X, \mathcal{E}_X)$ and $Y : (\Omega, \mathcal{F}, \mathbb{P}) \to (E_Y, \mathcal{E}_Y)$. Next, we define the following measures. Let $\mu_X(A) := \mathbb{P}(X^{-1}(A))$ and $\mu_Y(A) := \mathbb{P}(Y^{-1}(A))$ be *measures* induced by X and Y on $(E_X, \mathcal{E}_X)$ and $(E_Y, \mathcal{E}_Y)$, respectively. Furthermore, let $\mu_{X,Y}$ and $\mu_X \times \mu_Y$ be the *joint* and *product* measure, defined by $\mu_{X,Y}(A) := \mathbb{P}(\{\omega \in \Omega | (X(\omega), Y(\omega)) \in A\})$ and $(\mu_X \times \mu_Y)(A \times B) := \mu_X(A)\mu_Y(B)$ on $(E_X \times E_Y, \mathcal{E}_X \bigotimes \mathcal{E}_Y)$, respectively.

6.A Formulations of expected absolute change in distribution

In this appendix, we give multiple formulations of the *expected absolute change in distribution* (UD). Depending on the type of RV's, these formulations can be used.

6.A.1 General case:

For any X, Y the UD is defined as

$$\begin{aligned}\mathrm{UD}(X,Y) := &\sup_{A\in\mathcal{E}(X)\bigotimes\mathcal{E}(Y)} \left\{\mu_{(X,Y)}(A) - (\mu_X\times\mu_Y)(A)\right\} \\ &+ \sup_{B\in\mathcal{E}(X)\bigotimes\mathcal{E}(Y)} \left\{(\mu_X\times\mu_Y)(B) - \mu_{(X,Y)}(B)\right\} \\ = 2 &\sup_{A\in\mathcal{E}(X)\bigotimes\mathcal{E}(Y)} \left\{\mu_{(X,Y)}(A) - (\mu_X\times\mu_Y)(A)\right\}. \end{aligned} \tag{6.3}$$

6.A.2 Discrete RV's only:

When X, Y are discrete RV's, Equation (6.3) simplifies into

$$\mathrm{UD}(X,Y) := \sum_{x,y} \left|p_{X,Y}(x,y) - p_X(x)\cdot p_Y(y)\right|,$$

or equivalently

$$\mathrm{UD}(X,Y) := \sum_{x} p_X(x)\cdot \sum_{y} \left|p_{Y|X=x}(y) - p_Y(y)\right|.$$

Similarly, when X and Y take values in E_X and E_Y, respectively, Equation (6.3) becomes

$$\begin{aligned}\mathrm{UD}(X,Y) := &\sup_{A\subset E_X\times E_Y} \left\{\sum_{(x,y)\in A} (p_{X,Y}(x,y) - p_X(x)p_Y(y))\right\} \\ &+ \sup_{A\subset E_X\times E_Y} \left\{\sum_{(x,y)\in A} (p_X(x)p_Y(y) - p_{X,Y}(x,y))\right\} \\ = 2 &\sup_{A\subset E_X\times E_Y} \left\{\sum_{(x,y)\in A} (p_{X,Y}(x,y) - p_X(x)p_Y(y))\right\}. \end{aligned}$$

6.A.3 Continuous RV's only:

When X, Y are continuous RV's, Equation (6.3) becomes

$$\mathrm{UD}\,(X,Y) := \int_{\mathbb{R}}\int_{\mathbb{R}} |f_{X,Y}(x,y) - f_X(x)f_Y(y)|dydx,$$

or equivalently

$$\mathrm{UD}\,(X,Y) := \int_{\mathbb{R}} f_X(x) \int_{\mathbb{R}} |f_{Y|X=x}(y) - f_Y(y)|dydx.$$

Another formulation (more measure theoretical) would be:

$$\mathrm{UD}\,(X,Y) := 2 \cdot \sup_{A\in\mathcal{B}(\mathbb{R}^2)} \left\{ \int_A (f_{X,Y}(x,y) - f_X(x)f_Y(y))dydx \right\}.$$

6.A.4 Mix of discrete and continuous:

When X is discrete and Y is continuous, Equation (6.3) reduces to

$$\mathrm{UD}\,(X,Y) := \sum_x p_X(x) \int_y |f_{Y|X=x}(y) - f_Y(y)|dy.$$

Vice versa, if X is continuous and Y is discrete, Equation (6.3) becomes

$$\mathrm{UD}\,(X,Y) := \int_x f_X(x) \sum_y |p_{Y|X=x}(y) - p_Y(y)|dx.$$

6.B Properties of expected absolute change in distribution

In this appendix, we prove properties of the function UD that are used in Section 6.5 to show that the *BP* dependency measure satisfies all properties in Table 6.2.2.

6.B.1 Symmetry UD:

For the proofs below it is useful to show that $\mathrm{UD}(X,Y)$ is symmetric i.e. $\mathrm{UD}(X,Y) = \mathrm{UD}(Y,X)$ for every X, Y.

It directly follows from the definition, as

$$\begin{aligned}\mathrm{UD}(X,Y) &= 2 \sup_{A \in \mathcal{E}_X \bigotimes \mathcal{E}_Y} \left\{\mu_{(X,Y)}(A) - (\mu_X \times \mu_Y)(A)\right\} \\ &= 2 \sup_{A \in \mathcal{E}_Y \bigotimes \mathcal{E}_X} \left\{\mu_{(Y,X)}(A) - (\mu_Y \times \mu_X)(A)\right\} \\ &= \mathrm{UD}(Y,X).\end{aligned}$$

6.B.2 Independence and UD = 0:

Since we are considering a measure of dependence it is useful to know what the conditions for independence are. Below we will show that we have independence of X and Y if and only if $\mathrm{UD}(X,Y) = 0$.

Note that

$$\begin{aligned}\mathrm{UD}(X,Y) = &\sup_{A \in \mathcal{E}_X \bigotimes \mathcal{E}_Y} \left\{\mu_{(X,Y)}(A) - (\mu_X \times \mu_Y)(A)\right\} \\ &+ \sup_{B \in \mathcal{E}_X \bigotimes \mathcal{E}_Y} \left\{(\mu_X \times \mu_Y)(B) - \mu_{(X,Y)}(B)\right\} \\ \geq\ &\left(\mu_{(X,Y)}(E_X \times E_Y) - (\mu_X \times \mu_Y)(E_X \times E_Y)\right) \\ &+ \left((\mu_X \times \mu_Y)(E_X \times E_Y) - \mu_{(X,Y)}(E_X \times E_Y)\right) \\ = 0,\end{aligned}$$

with equality if and only if $\mu_{(X,Y)} = \mu_X \times \mu_Y$ on $\mathcal{E}_X \bigotimes \mathcal{E}_Y$, so if and only if X and Y are independent. So in conclusion properties i) and ii) below are equivalent:

(i) X and Y are independent random variables,

(ii) $\mathrm{UD}(X,Y) = 0$.

6.B.3 Upper bound for a given Y:

To scale the dependency function it is useful to know what the range of $\mathrm{UD}(X,Y)$ is for a given random variable Y. We already know it is lower bounded by 0 (see Section 6.B.2). However, we have not yet established an upper bound. What follows down below is a derivation of the upper bound.

A μ_Y-atom A is a set such that $\mu_Y(A) > 0$ and for any $B \subset A$ we have $\mu_Y(B) \in \{0, \mu_Y(A)\}$. Consider the following equivalence relation $\sim$ on the μ_Y-atoms characterized by $S \sim T$ if and only if $\mu_Y(S \triangle T) = 0$. Then let I be a set containing exactly one representative from each equivalence class. Note that I is countable, so we can enumerate the elements $A_1, A_2, A_3 \ldots$. Additionally, for any $A, B \in I$ we have that $\mu_Y(A \cap B) = 0$.

Next, we define $B_i := A_i \setminus \bigcup_{j=1}^{i-1} A_j$ to obtain a set of disjoint μ_Y-atoms. In what follows we assume I to be infinite, but the proof works exactly the same for finite I when you replace ∞ with $|I|$.

Let $E_Y^* := E_Y \setminus \bigcup_{j=1}^{\infty} B_j$, so that the B_j's and the E_Y^* form a partition of E_Y. Furthermore, let $b_j := \mu_Y(B_j)$ be the probabilities of being in the individual atoms in I (and therefore the sizes corresponding to the equivalence classes of atoms). It now holds for any RV X that:

$$
\begin{aligned}
\mathrm{UD}(X,Y) &= 2 \sup_{A \in \mathcal{E}_X \otimes \mathcal{E}_Y} \{\mu_{X,Y}(A) - (\mu_X \times \mu_Y)(A)\} \\
&\leq 2 \sup_{A \in \mathcal{E}_X \otimes \mathcal{E}_Y} \Big\{ \mu_{X,Y}\left(A \cap (E_X \times E_Y^*)\right) \\
&\qquad - (\mu_X \times \mu_Y)\left(A \cap (E_X \times E_Y^*)\right) \Big\} \\
&\quad + 2 \sup_{A \in \mathcal{E}_X \otimes \mathcal{E}_Y} \Big\{ \sum_{j=1}^{\infty} \Big(\mu_{X,Y}\left(A \cap (E_X \times B_j)\right) \\
&\qquad - (\mu_X \times \mu_Y)\left(A \cap (E_X \times B_j)\right) \Big) \Big\}.
\end{aligned} \tag{6.4}
$$

Now note that the first term is at most $\mu_Y(E_Y^*) = 1 - \sum_{i=1}^{\infty} b_i$. To bound

the second term, we examine each individual term of the summation. First we note that the set of finite unions of 'rectangles' (Cartesian products of elements in $\mathcal{E}_X$ and $\mathcal{E}_Y$):

$$R := \left\{ C \in \mathcal{E}_X \bigotimes \mathcal{E}_\mathcal{Y} \;\middle|\; \exists k \in \mathbb{N} \text{ s.t. } C = \bigcup_{i=1}^{k} (A_i \times B_i) \right. \\ \left. \text{with } \forall i : A_i \in \mathcal{E}_X \wedge B_i \in \mathcal{E}_Y \right\}$$

is an algebra. Therefore, for any $D \in \mathcal{E}_X \bigotimes \mathcal{E}_Y$ and $\epsilon > 0$, there exists a $D_\epsilon \in R$ such that $\nu(D_\epsilon \triangle D) < \epsilon$, where $\nu := \mu_{X,Y} + (\mu_X \times \mu_Y)$. Specifically for $A \cap (E_X \times B_j)$ and $\epsilon > 0$, there exists a $B_{j,\epsilon} \in R$ such that $\nu(B_{j,\epsilon} \triangle A \cap (E_X \times B_j)) < \epsilon$ and $B_{j,\epsilon} \subset E_X \times B_j$ holds, since intersecting with this set only decreases the expression whilst remaining in R.

Thus, we have that

$$|\mu_{X,Y}(A \cap (E_X \times B_j)) - \mu_{X,Y}(B_{j,\epsilon})| \\ + |(\mu_X \times \mu_Y)(A \cap (E_X \times B_j)) - (\mu_X \times \mu_Y)(B_{j,\epsilon})| < \epsilon.$$

Therefore, it must hold that

$$\mu_{X,Y}(A \cap (E_X \times B_j)) - (\mu_X \times \mu_Y)(A \cap (E_X \times B_j)) \\ \leq \mu_{X,Y}(B_{j,\epsilon}) - (\mu_X \times \mu_Y)(B_{j,\epsilon}) + \epsilon. \quad (6.5)$$

Since $B_{j,\epsilon}$ is a finite union of 'rectangles', we can also write it as a finite union of k disjoint 'rectangles such that $B_{j,\epsilon} = \bigcup_{i=1}^{k} S_i \times T_i$ with $S_i \in \mathcal{E}_X$ and $T_i \in \mathcal{E}_Y$ for all i. The upper bound in Equation (6.5) without ϵ can now be written as

$$\mu_{X,Y}(B_{j,\epsilon}) - (\mu_X \times \mu_Y)(B_{j,\epsilon}) = \sum_{i=1}^{k} \left(\mu_{X,Y}(S_i \times T_i) - (\mu_X \times \mu_Y)(S_i \times T_i)\right).$$

For all i it holds that $T_i \subset B_j$ which means that either $\mu_Y(T_i) = 0$ or $\mu_Y(T_i) = b_j$, since B_j is an atom of size b_j. This allows us to separate the

sum

$$\sum_{i=1}^{k} (\mu_{X,Y}(S_i \times T_i) - (\mu_X \times \mu_Y)(S_i \times T_i))$$
$$= \sum_{i:\mu_Y(T_i)=0} (\mu_{X,Y}(S_i \times T_i) - (\mu_X(S_i) \times \mu_Y(T_i))$$
$$+ \sum_{i:\mu_Y(T_i)=b_j} (\mu_{X,Y}(S_i \times T_i) - (\mu_X(S_i) \times \mu_Y(T_i)) = \star.$$

The first sum is equal to zero, since $\mu_{X,Y}(S_i \times T_i) \leq \mu_Y(T_i) = 0$. The second sum is upper bounded by $\mu_{X,Y}(S_i \times T_i) \leq \mu_{X,Y}(S_i \times B_j)$. By defining $S' = \bigcup_{i:\mu_Y(T_i)=b_j} S_i$, we obtain

$$\star \leq 0 + \sum_{i:\mu_Y(T_i)=b_j} (\mu_{X,Y}(S_i \times B_j) - b_j \cdot \mu_X(S_i))$$
$$= \mu_{X,Y}(S' \times B_j) - b_j \cdot \mu_x(S')$$
$$\leq \min\left\{(1-b_j) \cdot \mu_X(S'), b_j \cdot (1-\mu_X(S'))\right\}$$
$$\leq b_j - b_j^2.$$

But, since this is true for any $\epsilon > 0$, it holds that Equation (6.5) becomes

$$\mu_{X,Y}(A \cap (E_X \times B_j)) - (\mu_X \times \mu_Y)(A \cap (E_X \times B_j)) \leq b_j - b_j^2.$$

Plugging this back into Equation (6.4) gives

$$\begin{aligned}\mathrm{UD}(X,Y) \leq 2 \sup_{A\in\mathcal{E}_X\bigotimes\mathcal{E}_Y} \Big\{ & \mu_{X,Y}(A\cap(E_X\times E_Y^*)) \\ & - (\mu_X\times\mu_Y)(A\cap(E_X\times E_Y^*))\Big\} \\ + 2 \sup_{A\in\mathcal{E}_X\bigotimes\mathcal{E}_Y} \Bigg\{ & \sum_{j=1}^{\infty}\Big(\mu_{X,Y}(A\cap(E_X\times B_j)) \\ & - (\mu_X\times\mu_Y)(A\cap(E_X\times B_j))\Big)\Bigg\}\end{aligned}$$

$$\begin{aligned}&\leq 2\left(1-\sum_{i=1}^{\infty} b_i\right) + 2\cdot\sum_{j=1}^{\infty}(b_j - b_j^2) \\ &= 2\left(1-\sum_{i=1}^{\infty} b_i^2\right).\end{aligned}$$

Note that in the *continuous* case the summation is equal to 0. The upper bound simply becomes 2. In the *discrete* case, where E_Y is the set in which Y takes it's values, the expression becomes

$$\mathrm{UD}(X,Y) \leq 2\left(1-\sum_{i\in E_Y}\mathbb{P}(Y=i)^2\right).$$

6.B.4 Functional dependence attains maximum UD:

Since we established an upper bound in Section 6.B.3, the next step is to check whether this bound is actually *attainable*. What follows is a proof that this bound is achieved for any random variable X for which it holds that $Y = f(X)$ for some measurable function f.

Let $Y = f(X)$ for some measurable function f, then $\mu_X(f^{-1}(C)) = \mu_Y(C)$ for all $C \in \mathcal{E}_Y$. Let the μ_Y-atoms B_j and E_Y^* be the same as in Section 6.B.3. Since E_Y^* contains no atoms, for every $\epsilon > 0$ there exists a partition $T_1, \ldots, T_k$ for some $k \in \mathbb{N}$ such that $\mu_Y(T_i) < \epsilon$ for all $i \in \{1, \ldots, k\}$. Next, consider the

set $K = \bigcup_i \left(f^{-1}(T_i) \times T_i\right) \cup \bigcup_j \left(f^{-1}(B_j) \times B_j\right)$. It now follows that

$$\begin{aligned}
\mathrm{UD}\,(X,Y) &= 2 \sup_{A \in \mathcal{E}_Y \bigotimes \mathcal{E}_Y} \{\mu_{X,Y}(A) - (\mu_X \times \mu_Y)(A)\} \\
&\geq 2\left(\mu_{X,Y}(K) - (\mu_X \times \mu_Y)(K)\right) \\
&= 2\Bigg(\sum_i \left(\mu_{X,Y}\left(f^{-1}(T_i) \times T_i\right) - \mu_X\left(f^{-1}(T_i)\right)\mu_Y\left(T_i\right)\right) \\
&\qquad + \sum_j \left(\mu_{X,Y}\left(f^{-1}(B_j) \times B_j\right) - \mu_X\left(f^{-1}(B_j)\right)\mu_Y(B_j)\right)\Bigg) \\
&\geq 2\left(\sum_i \left(\mu_Y(T_i) - \epsilon \cdot \mu_Y\left(T_i\right)\right) + \sum_j \left(b_j - b_j^2\right)\right) \\
&= 2\left(\left(1 - \sum_j b_j\right) - \epsilon \cdot \left(1 - \sum_j b_j\right) + \sum_j \left(b_j - b_j^2\right)\right).
\end{aligned}$$

But, since this holds for any $\epsilon > 0$ we have

$$\mathrm{UD}\,(X,Y) \geq 2 \cdot \left(1 - \sum_j b_j^2\right).$$

As this is also the upper bound from Section 6.B.3, equality must hold. Thus, we can conclude that $\mathrm{UD}\,(X,Y)$ is maximal for Y if $Y = f(X)$, in particular if $X = Y$. It also follows that for any measurable function f and RV's X_1, X_2, Y with $Y = f(X_1)$, it must hold that $\mathrm{UD}\,(X_1,Y) \geq \mathrm{UD}\,(X_2,Y)$. Note that a corollary of this proof is that $\mathrm{UD}\,(Y,Y) = 0$ if and only if there exists a μ_Y-atom B_i with $\mu_Y(B_i) = 1$, or in other words there are no events that occur with a probability strictly between 0 and 1.

6.B.5 Unambiguity:

In Section 6.5, we show for discrete RV's that property II.5 holds. In this section, we prove the general case. Let $Y_1, \ldots, Y_N$ and S be independent

RV's where S takes values in $1, \ldots, N$ with $\mathbb{P}(S = i) = p_i$. Finally, define $X := Y_S$. Then we will show that $\mathrm{Dep}\,(Y_i|X) = p_i$.

Let $\mathcal{E}$ be the σ-algebra on which the independent Y_i are defined. Then we have $\mu_{X,Y_i,S}(A \times \{j\}) = \mu_{Y_j,Y_i}(A) \cdot \mu_S(\{j\}) = p_j \cdot \mu_{Y_j,Y_i}(A)$ for all j. Additionally, we have $\mu_X(A) = \sum_j p_j \cdot \mu_{Y_j}(A)$. Lastly, due to independence for $i \neq j$ we have $\mu_{Y_j,Y_i} = \mu_{Y_j} \times \mu_{Y_i}$. Combining this, gives

$$\begin{aligned}
\mathrm{UD}\,(X, Y_i) &= 2 \sup_{A \in \mathcal{E} \times \mathcal{E}} \{\mu_{X,Y_i}(A) - (\mu_X \times \mu_{Y_i})(A)\} \\
&= 2 \sup_{A \in \mathcal{E} \times \mathcal{E}} \left\{ \sum_j \mu_{X,Y_i,S}(A \times \{j\}) - \sum_j p_j (\mu_{Y_j} \times \mu_{Y_i})(A) \right\} \\
&= 2 \sup_{A \in \mathcal{E} \times \mathcal{E}} \left\{ \sum_j p_j \left(\mu_{Y_j,Y_i}(A) - (\mu_{Y_j} \times \mu_{Y_i})(A) \right) \right\} \\
&= 2 \sup_{A \in \mathcal{E} \times \mathcal{E}} \{p_i \left(\mu_{Y_i,Y_i}(A) - (\mu_{Y_i} \times \mu_{Y_i})(A) \right)\} \\
&= p_i \cdot \mathrm{UD}\,(Y_i, Y_i)\,.
\end{aligned}$$

6.B.6 Measurable functions never increase UD:

Next, we prove another useful property of UD: applying a measurable function to one of the variables does not increase the UD. Let $f : (E_X, \mathcal{E}_X) \to (E_{X'}, \mathcal{E}_{X'})$ be a measurable function. Note that $h : E_X \times E_Y \to E_{X'} \times E_Y$ with $h(x, y) := (f(x), y)$ is measurable. Now it follows that

$$\begin{aligned}
\mathrm{UD}\,(f(X), Y) &= 2 \sup_{A \in \mathcal{E}_{X'} \bigotimes \mathcal{E}_Y} \left\{ \mu_{(f(X),Y)}(A) - (\mu_{f(X)} \times \mu_Y)(A) \right\} \\
&= 2 \sup_{A \in \mathcal{E}_{X'} \bigotimes \mathcal{E}_Y} \left\{ \mu_{(X,Y)}(h^{-1}(A)) - (\mu_X \times \mu_Y)(h^{-1}(A)) \right\},
\end{aligned}$$

with $h^{-1}(A) \in \mathcal{E}_X \bigotimes \mathcal{E}_Y$. Thus,

$$\begin{aligned}
\mathrm{UD}\,(f(X), Y) &\leq 2 \sup_{A \in \mathcal{E}_X \bigotimes \mathcal{E}_Y} \left\{ \mu_{(X,Y)}(A) - (\mu_X \times \mu_Y)(A) \right\} \\
&= \mathrm{UD}\,(X, Y)\,.
\end{aligned}$$

In Section 6.B.1, it is proven that UD is symmetric. Therefore, it also holds for $g : E_Y \to E_{Y'}$, that

$$\mathrm{UD}\left(X, g(Y)\right) \leq \mathrm{UD}\left(X, Y\right).$$

Chapter 7

The Berkelmans-Pries Feature Importance Method: A Generic Measure of Informativeness of Features

Contents

Based on *Joris Pries, Guus Berkelmans, Sandjai Bhulai, and Rob van der Mei* (2023): "**The Berkelmans-Pries Feature Importance method: A generic measure of informativeness of features**". Submitted for publication. [135]

Abstract

Over the past few years, the use of machine learning models has emerged as a generic and powerful means for prediction purposes. At the same time, there is a growing demand for *interpretability* of prediction models. To determine which features of a dataset are important to predict a target variable Y, a *Feature Importance* (FI) method can be used. By quantifying how important each feature is for predicting Y, irrelevant features can be identified and removed, which could increase the speed and accuracy of a model, and moreover, important features can be discovered, which could lead to valuable insights. A major problem with evaluating FI methods, is that the ground truth FI is often unknown. As a consequence, existing FI methods do not give the exact correct FI values. This is one of the many reasons why it can be hard to properly interpret the results of an FI method. Motivated by this, we introduce a new global approach named the *Berkelmans-Pries* FI method, which is based on a combination of *Shapley values* and the *Berkelmans-Pries* dependency function. We prove that our method has many useful properties, and accurately predicts the correct FI values for several cases where the ground truth FI can be derived in an exact manner. We experimentally show for a large collection of FI methods (468) that existing methods do not have the same useful properties. This shows that the Berkelmans-Pries FI method is a highly valuable tool for analyzing datasets with complex interdependencies.

7.1 Introduction

How important are you? This is a question that researchers (especially data scientists) have wondered for many years. Researchers need to understand how important a random variable (RV) X is for determining Y. Which features are important for predicting the weather? Can indicators be found as symptoms for a specific disease? Can redundant variables be discarded to increase performance? These kinds of questions are relevant in almost any research area. Especially nowadays, as the rise of machine learning models generates the need to demystify prediction models. Altmann et al. [6] state that "In life sciences, interpretability of machine learning models is as important as their prediction accuracy." Although this might not hold for all research areas, interpretability is very useful. Knowing how predictions are made and why, is crucial for adapting these methods in everyday life.

Determining *Feature Importance* (FI) is the art of discovering the importance of each feature X_i when predicting Y. The following two cases are particularly useful. (I) Finding variables that are not important: *redundant* variables can be discovered using FI methods. Irrelevant features could degrade the performance of a prediction model due to high dimensionality and irrelevant information [89]. Eliminating redundant features could therefore increase both the speed and the accuracy of a prediction model. (II) Finding variables that are important: *important* features could reveal underlying structures that give valuable insights. Observing that variable X is important for predicting Y could steer research efforts into the right direction. Although it is critical to keep in mind that high FI does not mean causation. However, FI values do, for example, "enable an anaesthesiologist to better formulate a diagnosis by knowing which attributes of the patient and procedure contributed to the current risk predicted" [111]. In this way, an FI method can have really meaningful impact.

Over the years, many FI methods have been suggested, which results in a wide range of FI values for the same dataset. For example, stochastic methods do not even repeatedly predict the same FI values. This makes interpretation difficult. Examine e.g., a result of Fryer et al. [55], where one measure assigns an FI of 3.19 to a variable, whereas another method gives the same variable an FI value of 0.265. This raises a lot of questions: 'Which FI method is correct?', 'Is this variable deemed important?', and more generally 'What information does this give us?'. To assess the performance of an FI method, the ground truth should be known, which is often not the case [2,

71, 176, 200]. Therefore, when FI methods were developed, the focus has not yet lied on predicting the *exact* correct FI values. Additionally, many FI methods do not have desirable properties. For example, two features that contain the same amount of information should get the same FI. We later show that this is often not the case.

To improve interpretability, we introduce a new FI method called *Berkelmans-Pries* FI method, which is based on *Shapley values* [158] and the *Berkelmans-Pries* dependency function [13]. Multiple existing methods already use Shapley values, which has been shown to give many nice properties. However, by *additionally* using the *Berkelmans-Pries* dependency function, even more useful properties are obtained. Notably, we prove that this approach accurately predicts the FI in some cases where the ground truth FI can be derived in an exact manner. By combining *Shapley values* and the *Berkelmans-Pries* dependency function a powerful FI method is created. This research is an important step forward for the field of FI, because of the following reasons:

- We introduce a new FI method;
- We prove multiple useful properties of this method;
- We provide some cases where the ground truth FI can be derived in an exact manner;
- We prove for these cases that our FI method accurately predicts the correct FI;
- We obtain the largest collection of existing FI methods;
- We test if these methods adhere to the same properties, which shows that no method comes close to fulfilling all the useful properties;
- We provide Python code to determine the FI values [133].

7.2 The Berkelmans-Pries Feature Importance

Kruskal [93] stated that "There are infinitely many possible measures of association, and it sometimes seems that almost as many have been proposed at one time or another." Although this quote was about dependency functions, it could just as well have been about FI methods. Over the years, many FI methods have been suggested, but it remains unclear which method should be used when and why [71]. In this section, we propose yet another new FI method named the *Berkelmans-Pries FI method* (BP-FI).

Although it is certainly subjective what it is that someone wants from an FI method, we show in Section 7.3 that BP-FI has many useful and intuitive properties. The BP-FI method is based on two key elements: (1) *Shapley values* and (2) the *Berkelmans-Pries dependency function*. We will discuss these components first to clarify how the BP-FI method works.

7.2.1 Shapley value approach

The *Shapley value* is a unique game-theoretical way to assign *value* to each *player* participating in a multiplayer *game* based on four axioms [158]. This concept is widely used in FI methods, as it can be naturally adapted to determine how *important* (value) each *feature* (player) is for *predicting a target variable* (game). Let N_{vars} be the number of features, then the Shapley value of feature i is defined by

$$\phi_i(v) = \sum_{S \subseteq \{1,\dots,N_{\text{vars}}\} \setminus \{i\}} \frac{|S|! \cdot (N_{\text{vars}} - |S| - 1)!}{N_{\text{vars}}!} \cdot (v(S \cup \{i\}) - v(S)) \,, \tag{7.1}$$

where $v(S)$ can be interpreted as the 'worth' of the coalition S [158]. The principle behind this formulation can also be explained in words: For every possible sequence of features up to feature i, the added value of feature i is the difference between the worth before it was included (i.e., $v(S)$) and after (i.e., $v(S \cup \{i\})$). Averaging these added values over all possible sequences of features gives the final Shapley value for feature i.

SHAP There are multiple existing FI methods that use Shapley values [43, 55, 110], which immediately ensures some useful properties. The most famous of these methods is SHAP [110]. This method is widely used for *local* explanations (see Section 7.4.1). To measure the local FI for a specific sample x and a prediction model f, the *conditional expectation* is used as characteristic function (i.e., v in Equation (7.1)). Let $x = (x_1, x_2, \dots, x_{N_{\text{vars}}})$, where x_i is the feature value of feature i, then SHAP FI values can be determined using:

$$v_x(S) := \mathbb{E}_z \left[f(z) | z_i = x_i \text{ for all } i \in S, \text{ where } z = (z_1, \dots, z_{N_{\text{vars}}}) \right] . \tag{7.2}$$

Observe that the characteristic function v_x is defined locally for each x. To get *global* FI values, an average can be taken over all local FI values. Our novel FI method uses a different characteristic function, namely the

Berkelmans-Pries dependency function. This leads to many additional useful properties. Furthermore, the focus of this research is not on *local* explanations, but *global* FI values.

7.2.2 Berkelmans-Pries dependency function

A new dependency measure, called the *Berkelmans-Pries* (BP) dependency function, was introduced in [13], which is used in the formulation of the BP-FI method. It is shown that the *BP* dependency function satisfies a list of desirable properties, whereas existing dependency measures did not. It has a measure-theoretical formulation, but this reduces to a simpler and more intuitive version when all variables are discrete [13]. We want to highlight this formulation to give some intuition behind the *BP* dependency function. It is given by

$$\mathrm{Dep}\left(Y|X\right) := \begin{cases} \frac{\mathrm{UD}(X,Y)}{\mathrm{UD}(Y,Y)} & \text{if } Y \text{ is not a.s. constant,} \\ \text{undefined} & \text{if } Y \text{ is a.s. constant,} \end{cases} \tag{7.3}$$

where (in the discrete case) it holds that

$$\mathrm{UD}\left(X,Y\right) := \sum_x p_X(x) \cdot \sum_y \left| p_{Y|X=x}(y) - p_Y(y) \right|. \tag{7.4}$$

The *BP* dependency measure can be interpreted in the following manner. The numerator is the expected absolute difference between the distribution of Y and the distribution of Y given X. If Y is highly dependent on X, the distribution changes as knowing X gives information about Y, whereas if Y is independent of X, there is no difference between these two distributions. The denominator is the maximal possible change in distribution of Y for any variable, which is used to standardize the dependency function. Note that the *BP* dependency function is *asymmetric*: $\mathrm{Dep}\left(Y|X\right)$ is the dependency of Y on X, not vice versa. Due to the many desirable properties, the *BP* dependency function is used for the BP-FI.

7.2.3 Berkelmans-Pries FI method

One crucial component of translating the game-theoretical approach of Shapley values to the domain of FI is choosing the function v in Equation (7.1). This function assigns for each set of features S a value $v(S)$ that characterizes the 'worth' of the set S. How this function is defined, has a critical impact

on the resulting FI. We choose to define the 'worth' of a set S to be the *BP* dependency of Y on the set S, which is denoted by $\mathrm{Dep}(Y|S)$ [13]. Here, $\mathrm{Dep}(Y|S) = \mathrm{Dep}(Y|Z_S(\mathcal{D}))$ where $\mathcal{D}$ denotes the entire dataset with all features and $Z_S(\mathcal{D})$ is the reduction of the dataset to include only the subset of features S. Let Ω_{feat} be the set of all feature variables. Now, for every $S \subseteq \Omega_{\text{feat}}$, we define:

$$v(S) := \mathrm{Dep}(Y|S)\,. \tag{7.5}$$

In other words, the value of set S is exactly how *dependent* the target variable Y is on the features in S. The difference $v(S \cup \{i\}) - v(S)$ in Equation (7.1) can now be viewed as the increase in dependency of Y on the set of features, when feature i is also known. The resulting Shapley values using the *BP* dependency function as characteristic function are defined to be the BP-FI outcome. For each feature i, we get:

$$\begin{aligned}\mathrm{FI}(i) &:= \sum_{S \subseteq \Omega_{\text{feat}} \setminus \{i\}} \frac{|S|! \cdot (N_{\text{vars}} - |S| - 1)!}{N_{\text{vars}}!} \cdot (v(S \cup \{i\}) - v(S)) \\ &= \sum_{S \subseteq \Omega_{\text{feat}} \setminus \{i\}} \frac{|S|! \cdot (N_{\text{vars}} - |S| - 1)!}{N_{\text{vars}}!} \cdot (\mathrm{Dep}(Y|S \cup \{i\}) - \mathrm{Dep}(Y|S))\,.\end{aligned} \tag{7.6}$$

Abbreviated notation improves readability of upcoming derivations, which is why we define

$$w(S, N_{\text{vars}}) := \frac{|S|! \cdot (N_{\text{vars}} - |S| - 1)!}{N_{\text{vars}}!}, \tag{N1}$$

$$D(X, Y, S) := \mathrm{Dep}(Y|S \cup \{X\}) - \mathrm{Dep}(Y|S)\,. \tag{N2}$$

Note that when Y is *almost surely constant* (i.e., $\mathbb{P}(Y = y) = 1$), $\mathrm{Dep}(Y|S)$ is undefined for any feature set S (see Equation (7.3)). We argue that it is natural to assume that $\mathrm{FI}(i)$ is also undefined, as every feature attributes everything and nothing at the same time. In the remainder of this chapter, we assume that Y is not a.s. constant.

7.3 Properties of the Berkelmans-Pries Feature Importance

Recall that it is hard to evaluate FI methods, as the ground truth FI is often unknown [2, 71, 176, 200]. With this in mind, we want to show that the BP-FI method has many desirable properties. We also give some synthetic cases where the BP-FI method gives a natural expected outcome. The BP-FI method is stooled on *Shapley values*, which are a unique solution based on four axioms [188]. These axioms already give many characteristics that are preferable for an FI method. Additionally, using the *BP* dependency function ensures that it has extra desirable properties. In this section, we prove properties of the BP-FI method and discuss why these are relevant and useful.

Property 1 (Efficiency)**.** The sum of all FI scores is equal to the total dependency of Y on all features:

$$\sum_{i \in \Omega_{\text{feat}}} \text{FI}(i) = \text{Dep}\left(Y | \Omega_{\text{feat}}\right).$$

Proof. Shapley values are *efficient*, meaning that all the value is distributed among the players. Thus,

$$\sum_{i \in \Omega_{\text{feat}}} \text{FI}(i) = v(\Omega_{\text{feat}}) = \text{Dep}\left(Y | \Omega_{\text{feat}}\right). \qquad \square$$

Relevance. With our approach, we try to answer the question 'How much did each feature contribute to the total dependency?'. The total 'payoff' is in our case the total dependency. It is therefore natural to divide the entire payoff (but not more than that) amongst all features.

Corollary 1.1. If adding a RV X to the dataset does not give any additional information (i.e., $\text{Dep}\left(Y | \Omega_{\text{feat}} \cup X\right) = \text{Dep}\left(Y | \Omega_{\text{feat}}\right)$), then the sum of all FI remains the same.

Proof. This directly follows from Property 1. $\square$

Relevance. If the collective knowledge remains the same, the same amount of credit is available to be divided amongst the features. Only when new information is added, an increase in combined credit is warranted. A direct result of this corollary is that adding a *clone* (i.e., $X^{\text{clone}} := X$) of a variable X to the dataset will never increase the total sum of FI.

Property 2 (Symmetry)**.** When it holds for every $S \subseteq \Omega_{\text{feat}} \setminus \{i, j\}$ that $\text{Dep}(Y|S \cup \{i\}) = \text{Dep}(Y|S \cup \{j\})$, then $\text{FI}(i) = \text{FI}(j)$.

Proof. Shapley values are *symmetric*, meaning that if $v(S \cup \{i\}) = v(S \cup \{j\})$ for every $S \subseteq \Omega_{\text{feat}} \setminus \{i, j\}$, it follows that $\text{FI}(i) = \text{FI}(j)$. Thus, it automatically follows that BP-FI is also symmetric. □

Relevance. If two variables are interchangeable, meaning that they *always* contribute equally to the dependency, it is only sensible that they obtain the same FI. This is a desirable property for an FI method, as two features that contribute equally should obtain the same FI.

Property 3 (Range)**.** For any RV X, it holds that $\text{FI}(X) \in [0, 1]$.

Proof. The *BP* dependency function is *non-increasing* under functions of X [13], which means that for any measurable function f it holds that

$$\text{Dep}(Y|f(X)) \leq \text{Dep}(Y|X)\,.$$

Take $f := Z_S$, which is the function that reduces $\mathcal{D}$ to the subset of features in S. Using the non-increasing property of *BP* dependency function, it follows that:

$$\begin{aligned} \text{Dep}(Y|S) = \text{Dep}(Y|Z_S(\mathcal{D})) = \text{Dep}\left(Y|Z_S(Z_{S\cup\{i\}}(\mathcal{D}))\right) \\ \leq \text{Dep}\left(Y|Z_{S\cup\{i\}}(\mathcal{D})\right) = \text{Dep}(Y|S \cup \{i\})\,. \end{aligned} \tag{7.7}$$

Examining Equation (7.6), we observe that every FI value must be greater or equal to zero, as $\text{Dep}(Y|S \cup \{i\}) - \text{Dep}(Y|S) \geq 0$.

One of the properties of the *BP* dependency function is that for any X, Y it holds that $\text{Dep}(Y|X) \in [0, 1]$ [13]. Using Property 1, the sum of all FI values must therefore be in $[0, 1]$, as $\sum_{i\in\Omega_{\text{feat}}} \text{FI}(i) = \text{Dep}(Y|\Omega_{\text{feat}}) \in [0, 1]$. This gives an upper bound for the FI values, which is why we can now conclude that $\text{FI}(X) \in [0, 1]$ for any RV X. □

Relevance. It is essential for interpretability that an FI method is bounded by known bounds. For example, an FI score of 4.2 cannot be interpreted properly, when the upper or lower bound is unknown.

Property 4 (Bounds)**.** Every $\text{FI}(X)$ with $X \in \Omega_{\text{feat}}$ is bounded by

$$\frac{\text{Dep}(Y|X)}{N_{\text{vars}}} \leq \text{FI}(X) \leq \text{Dep}(Y|\Omega_{\text{feat}})\,.$$

Proof. The upper bound follows from Properties 1 and 3, as

$$\mathrm{Dep}\,(Y|\Omega_{\mathrm{feat}}) = \sum_{i\in\Omega_{\mathrm{feat}}} \mathrm{FI}(i) \geq \mathrm{FI}(X),$$

where the last inequality follows since $\mathrm{FI}(i) \in [0,1]$ for all $i \in \Omega_{\mathrm{feat}}$.

The lower bound can be established using the inequality from Equation (7.7) within Equation (7.6). This gives (using Notation (N1))

$$\begin{aligned}
\mathrm{FI}(X) &= \sum_{S\subseteq\Omega_{\mathrm{feat}}\setminus\{X\}} w(S, N_{\mathrm{vars}}) \cdot \Big(\mathrm{Dep}\,(Y|S\cup\{X\}) - \mathrm{Dep}\,(Y|S)\Big) \\
&\geq w(0, N_{\mathrm{vars}}) \cdot (\mathrm{Dep}\,(Y|\varnothing\cup\{X\}) - \mathrm{Dep}\,(Y|\varnothing)) \\
&= \frac{0!\cdot(N_{\mathrm{vars}}-0-1)!}{N_{\mathrm{vars}}!} \cdot \mathrm{Dep}\,(Y|X) \\
&= \frac{\mathrm{Dep}\,(Y|X)}{N_{\mathrm{vars}}}.
\end{aligned}$$

□

Relevance. These bounds are useful for upcoming proofs.

Property 5 (Zero FI)**.** For any RV X, it holds that

$$\mathrm{FI}(X) = 0 \Leftrightarrow \mathrm{Dep}\,(Y|S\cup\{X\}) = \mathrm{Dep}\,(Y|S) \text{ for all } S \in \Omega_{\mathrm{feat}}\setminus\{X\}.$$

Proof. $\Leftarrow$: When $\mathrm{Dep}\,(Y|S\cup\{X\}) = \mathrm{Dep}\,(Y|S)$ for all $S \in \Omega_{\mathrm{feat}}\setminus\{X\}$, it immediately follows from Equation (7.6) (with Notation (N1)) that

$$\begin{aligned}
\mathrm{FI}(X) &= \sum_{S\subseteq\Omega_{\mathrm{feat}}\setminus\{X\}} w(S, N_{\mathrm{vars}}) \cdot \Big(\mathrm{Dep}\,(Y|S\cup\{X\}) - \mathrm{Dep}\,(Y|S)\Big) \\
&= \sum_{S\subseteq\Omega_{\mathrm{feat}}\setminus\{X\}} \frac{|S|!\cdot(N_{\mathrm{vars}}-|S|-1)!}{N_{\mathrm{vars}}!} \cdot 0 \\
&= 0.
\end{aligned}$$

$\Rightarrow$: Assume that $\mathrm{FI}(X) = 0$ holds. It follows from the proof of Property 3 that $\mathrm{Dep}\,(Y|S\cup\{X\}) - \mathrm{Dep}\,(Y|S) \geq 0$ for every $S \subseteq \Omega_{\mathrm{feat}}\setminus\{X\}$. Let

$S^* \in \Omega_{\text{feat}} \setminus \{X\}$ be given such that $\text{Dep}\,(Y|S^* \cup \{X\}) - \text{Dep}\,(Y|S^*) > 0$. It follows from Equation (7.6) (with Notation (N1)) that

$$\begin{aligned}
\text{FI}(X) &= \sum_{S \subseteq \Omega_{\text{feat}} \setminus \{X\}} w(S, N_{\text{vars}}) \cdot \big(\text{Dep}\,(Y|S \cup \{X\}) - \text{Dep}\,(Y|S)\big) \\
&\geq w(S^*, N_{\text{vars}}) \cdot (\text{Dep}\,(Y|S^* \cup \{X\}) - \text{Dep}\,(Y|S^*)) \\
&= \frac{|S^*|! \cdot (N_{\text{vars}} - |S^*| - 1)!}{N_{\text{vars}}!} \cdot (\text{Dep}\,(Y|S^* \cup \{X\}) - \text{Dep}\,(Y|S^*)) \\
&> 0.
\end{aligned}$$

This gives a contradiction with the assumption that $\text{FI}(X) = 0$, thus it is not possible that such an S^* exists. This means that $\text{Dep}\,(Y|S \cup \{X\}) = \text{Dep}\,(Y|S)$ for all $S \in \Omega_{\text{feat}} \setminus \{X\}$. □

Relevance. When a feature *never* contributes any information, it is only fair that it does not receive any FI. The feature can be removed from the dataset, as it has no effect on the target variable. On the other hand, when a feature has an FI of zero, it would be unfair to this feature if it does in fact contribute information somewhere. It should then be rewarded some FI, albeit small it should be larger than zero.

Null-independence The property that a feature receives zero FI, when $\text{Dep}\,(Y|S \cup \{X\}) = \text{Dep}\,(Y|S)$ for all $S \in \Omega_{\text{feat}} \setminus \{X\}$, is the same notion as a *null player* in game theory. Berkelmans et al. [13] show that $\text{Dep}\,(Y|X) = 0$, when Y is *independent* of X. To be a *null player* requires a stricter definition of independence, which we call *null-independence*. Y is null-independent on X if $\text{Dep}\,(Y|S \cup \{X\}) = \text{Dep}\,(Y|S)$ for all $S \in \Omega_{\text{feat}} \setminus \{X\}$. In other words, X is null-independent if and only if $\text{FI}(X) = 0$.

Corollary 5.1. Independent feature $\not\Rightarrow$ null-independent feature.

Proof. Take e.g., the dataset consisting of two binary features $X_1, X_2 \sim \mathcal{U}(\{0,1\})$ and a target variable $Y = X_1 \cdot (1 - X_2) + X_2 \cdot (1 - X_1)$ which is the XOR of X_1 and X_2. Individually, the variables do not give any information about Y, whereas collectively they fully determine Y. In the proof of Property 15, we show that this leads to $\text{FI}(X_1) = \text{FI}(X_2) = \frac{1}{2}$, whilst $\text{Dep}\,(Y|X_1) = \text{Dep}\,(Y|X_2) = 0$. Thus, X_1 and X_2 are *independent*, but not *null-independent*. □

Corollary 5.2. Independent feature ⇐ null-independent feature.

Proof. When X is *null-independent*, it holds that $\mathrm{FI}(X) = 0$. Using Property 4, we obtain

$$0 = \mathrm{FI}(X) \geq \frac{\mathrm{Dep}\,(Y|X)}{N_{\mathrm{vars}}} \Leftrightarrow \mathrm{Dep}\,(Y|X) = 0.$$

Thus, when X is *null-independent*, it is also *independent*. □

Corollary 5.3. Almost surely constant variables get zero FI.

Proof. If X is *almost surely constant* (i.e., $\mathbb{P}(X = x) = 1$), it immediately follows that $\mathrm{Dep}\,(Y|S \cup \{X\}) = \mathrm{Dep}\,(Y|S)$ for any $S \subseteq \Omega_{\mathrm{feat}} \setminus \{X\}$, as the distribution of Y is not affected by X. □

Property 6 (FI equal to one)**.** When $\mathrm{FI}(X) = 1$, it holds that $\mathrm{Dep}\,(Y|X) = 1$ and all other features are null-independent.

Proof. As the *BP* dependency function is bounded by $[0, 1]$ [13], it follows from Property 1 that $\sum_{i \in \Omega_{\mathrm{feat}}} \mathrm{FI}(i) \leq 1$. Noting that each FI must be in $[0, 1]$ due to Property 3, we find that

$$\mathrm{FI}(X) = 1 \Rightarrow \mathrm{FI}(X') = 0 \text{ for all } X' \in \Omega_{\mathrm{feat}} \setminus \{X\}.$$

Thus, all other features have to be *null-independent*. Next, we show that $\mathrm{Dep}\,(Y|X) = 1$ must also hold, when $\mathrm{FI}(X) = 1$. Assume that $\mathrm{Dep}\,(Y|X) < 1$. Using Equation (7.6) (with Notations (N1) and (N2)) we find that

$$\begin{aligned}
1 = \mathrm{FI}(X) &= \sum_{S \subseteq \Omega_{\mathrm{feat}} \setminus \{X\}} w(S, N_{\mathrm{vars}}) \cdot D(X, Y, S) \\
&= \sum_{S \subseteq \Omega_{\mathrm{feat}} \setminus \{X\} : |S| > 0} (w(S, N_{\mathrm{vars}}) \cdot D(X, Y, S)) + w(\varnothing, N_{\mathrm{vars}}) \cdot D(X, Y, \varnothing) \\
&\leq \sum_{S \subseteq \Omega_{\mathrm{feat}} \setminus \{X\} : |S| > 0} (w(S, N_{\mathrm{vars}}) \cdot (1 - 0)) + w(\varnothing, N_{\mathrm{vars}}) \cdot (\mathrm{Dep}\,(Y|X) - 0) \\
&= \star
\end{aligned}$$

Note that the last step follows from the range of the *BP* dependency function (i.e., $[0, 1]$). The largest possible addition is when $\mathrm{Dep}\,(Y|S \cup \{X\}) - \mathrm{Dep}\,(Y|S) = 1 - 0 = 1$. We then get

$$
\begin{aligned}
\star &< \sum_{S \subseteq \Omega_{\text{feat}} \setminus \{X\}} w(S, N_{\text{vars}}) \\
&= \sum_{k=0}^{N_{\text{vars}}-1} \binom{N_{\text{vars}} - 1}{k} \cdot \frac{k! \cdot (N_{\text{vars}} - k - 1)!}{N_{\text{vars}}!} \\
&= \sum_{k=0}^{N_{\text{vars}}-1} \frac{(N_{\text{vars}} - 1)!}{k! \cdot (N_{\text{vars}} - 1 - k)!} \cdot \frac{k! \cdot (N_{\text{vars}} - k - 1)!}{N_{\text{vars}}!} \\
&= \sum_{k=0}^{N_{\text{vars}}-1} \frac{1}{N_{\text{vars}}} \\
&= 1.
\end{aligned}
$$

This result gives a contradiction, as $1 < 1$ cannot be true, which means that $\mathrm{Dep}\,(Y|X) = 1$. □

Relevance. When a variable gets an FI of one, the rest of the variables should be zero. Additionally, it should mean that this variable contains the necessary information to fully determine Y, which is why $\mathrm{Dep}\,(Y|X) = 1$ should hold.

Property 7. $\mathrm{Dep}\,(Y|X) = 1 \not\Rightarrow \mathrm{FI}(X) = 1$.

Proof. As counterexample, examine the case where there are multiple variables that fully determine Y. Properties 1 and 3 must still hold. Thus, if FI is one for every variable that fully determines Y, we get

$$
\sum_{i \in \Omega_{\text{feat}}} \mathrm{FI}(i) \geq 1 + 1 \neq 1 = \mathrm{Dep}\,(Y|\Omega_{\text{feat}})\,,
$$

which is a contradiction. □

Relevance. This property is important for interpretation of the FI score. When $\mathrm{FI}(X) \neq 1$, it cannot be automatically concluded that Y is not fully determined by X.

If Y is fully determined by X, we call X *fully informative*, as it gives all information that is necessary to determine Y.

Property 8 (Max FI when fully informative)**.** If X is fully informative, it holds that $\mathrm{FI}(i) \leq \mathrm{FI}(X)$ for any $i \in \Omega_{\mathrm{feat}}$.

Proof. Assume that there exists a feature i such that $\mathrm{FI}(i) > \mathrm{FI}(X)$, when Y is fully determined by X. To attain a higher FI, somewhere in the sum of Equation (7.6), a higher gain must be made by i compared to X. Observe that for any $S \subseteq \Omega_{\mathrm{feat}} \setminus \{i, X\}$ it holds that

$$\begin{aligned}\mathrm{Dep}\left(Y|S \cup \{i\}\right) - \mathrm{Dep}\left(Y|S\right) &\leq 1 - \mathrm{Dep}\left(Y|S\right) \\ &= \mathrm{Dep}\left(Y|S \cup \{X\}\right) - \mathrm{Dep}\left(Y|S\right).\end{aligned}$$

For any $S \subseteq \Omega_{\mathrm{feat}} \setminus \{i\}$ with $X \in S$, it holds that

$$\begin{aligned}\mathrm{Dep}\left(Y|S \cup \{i\}\right) - \mathrm{Dep}\left(Y|S\right) &= \mathrm{Dep}\left(Y|S \cup \{i\}\right) - 1 \\ &= 0.\end{aligned}$$

The last step follows from Equation (7.7), as the dependency function is increasing, thus $\mathrm{Dep}\left(Y|S \cup \{i\}\right) = 1$. In other words, no possible gain can be achieved with respect to X in the Shapley values. Therefore, it cannot hold that $\mathrm{FI}(i) > \mathrm{FI}(X)$. □

Relevance. Whenever a variable fully determines Y, it should attain the highest FI. What would an FI higher than such a score mean? It gives more information than the maximal information? When this property would not hold, it would result in a confusing and difficult interpretation process.

Property 9 (Limiting the outcome space)**.** For any measurable function f and RV X, replacing X with $f(X)$ never increases the assigned FI to this variable.

Proof. The *BP* dependency function is non-increasing under functions of X [13]. This means that for any measurable function g, it holds that

$$\mathrm{Dep}\left(Y|g(X)\right) \leq \mathrm{Dep}\left(Y|X\right).$$

Choose g to be the function that maps the union of any feature set S and the original RV X to the union of S and the replacement $f(X)$. In other

words $g(S \cup \{X\}) = S \cup \{f(X)\}$ for any feature set S. It then follows that:

$$\mathrm{Dep}\left(Y|S \cup \{f(X)\}\right) = \mathrm{Dep}\left(Y|g(S \cup \{X\})\right) \leq \mathrm{Dep}\left(Y|S \cup \{X\}\right),$$

and

$$\mathrm{Dep}\left(Y|S \cup \{f(X)\}\right) - \mathrm{Dep}\left(Y|S\right) \leq \mathrm{Dep}\left(Y|S \cup \{X\}\right) - \mathrm{Dep}\left(Y|S\right)$$

for any $S \subseteq \Omega_{\text{feat}} \setminus \{X\}$. Thus, using Equation (7.6), we can conclude that replacing X with $f(X)$ never increases the assigned FI. □

Relevance. This is an important observation for preprocessing. Whenever a variable is binned, it would receive less (or equal) FI when less bins are used. It could also potentially provide a useful upper bound, when the FI is already known before replacing X with $f(X)$.

Corollary 9.1. For any measurable function f and RV X, when $X = f(X')$ for another RV X', replacing feature X by feature X' will never decrease the assigned FI.

Proof. When $X = f(X')$ holds, it follows again (similar to Property 9) that

$$\mathrm{Dep}\left(Y|S \cup \{X\}\right) = \mathrm{Dep}\left(Y|S \cup \{f(X')\}\right) \leq \mathrm{Dep}\left(Y|S \cup \{X'\}\right)$$

for any $S \subseteq \Omega_{\text{feat}} \setminus \{X\}$. Therefore, using Equation (7.6), observe that replacing X with X' never decreases the assigned FI. □

Shapley values have additional properties when the characteristic function v is *subadditive* and/or *superadditive* [158]. We show that our function, defined by Equation (7.5), is neither.

Property 10 (Neither subadditive nor superadditive)**.** Our characteristic function $v(S) = \mathrm{Dep}\left(Y|S\right)$ is neither *subadditive* nor *superadditive*.

Proof. Consider the following two counterexamples.

Counterexample subadditive: A function f is *subadditive* if for any $S, T \in \Omega_{\text{feat}}$ it holds that

$$f(S \cup T) \leq f(S) + f(T).$$

Examine the dataset consisting of two binary features $X_1, X_2 \sim \mathcal{U}(\{0,1\})$ and a target variable $Y = X_1 \cdot (1 - X_2) + X_2 \cdot (1 - X_1)$ which is the XOR of

X_1 and X_2. Both X_1 and X_2 do not individually give any new information about the distribution of Y, thus $v(X_1) = v(X_2) = 0$ (see properties of the *BP* dependency function [13]). However, collectively they fully determine Y and thus $v(X_1 \cup X_2) = 1$. We can therefore conclude that v is not subadditive, as

$$v(X_1 \cup X_2) = 1 \not\leq 0 + 0 = v(X_1) + v(X_2).$$

Counterexample superadditive: A function f is *superadditive* if for any $S, T \in \Omega_{\text{feat}}$ it holds that

$$f(S \cup T) \geq f(S) + f(T).$$

Consider the dataset consisting of two binary features $X \sim \mathcal{U}(\{0,1\})$ and a *clone* $X^{\text{clone}} := X$, where the target variable Y is defined as $Y := X$. Note that both X and X^{clone} fully determine Y, thus $v(X) = v(X^{\text{clone}}) = 1$ (see properties of the *BP* dependency function [13]). Combining X and X^{clone} also fully determines Y, which leads to:

$$v(X \cup X^{\text{clone}}) = 1 \not\geq 1 + 1 = v(X) + v(X^{\text{clone}}).$$

Thus, v is also not superadditive. □

Relevance. If the characteristic function v is *subadditive*, it would hold that $\text{FI}(X) \leq v(X)$ for any $X \in \Omega_{\text{feat}}$. When v is *superadditive*, it follows that $\text{FI}(X) \geq v(X)$ for any $X \in \Omega_{\text{feat}}$. This is sometimes also referred to as *individual rationality*, which means that no player receives less, than what he could get on his own. This makes sense in a game-theoretic scenario with human players that can decide to not play when one could gain more by not cooperating. In our case, features do not have a free will, which makes this property not necessary. The above proof shows that v is in our case neither *subadditive* nor *superadditive*, which is why we cannot use their corresponding bounds.

Property 11 (Adding features can increase FI)**.** When an extra feature is added to the dataset, the FI of X can increase.

Proof. Consider the previously mentioned XOR dataset, where $X_1, X_2 \sim \mathcal{U}(\{0,1\})$ and $Y = X_1 \cdot (1 - X_2) + X_2 \cdot (1 - X_1)$. If at first, X_2 was not in the dataset, the FI of X_1 would be zero, as $\text{Dep}(Y|X_1) = 0$. However, if X_2 is added to the dataset, the FI of X_1 increases to $\frac{1}{2}$ (see Property 15). The FI of a feature can thus increase if another feature is added. □

Property 12 (Adding features can decrease FI)**.** When an extra feature is added to the dataset, the FI of X can decrease.

Proof. Consider the dataset given by $X \sim \mathcal{U}(\{0,1\})$ and $Y := X$. It immediately follows that $\mathrm{FI}(X) = 1$. However, when a *clone* is introduced ($X^{\mathrm{clone}} := X$), it holds that $\mathrm{FI}(X) = \mathrm{FI}(X^{\mathrm{clone}})$, because of Property 8. Additionally, it follows from Property 1 that $\mathrm{FI}(X) + \mathrm{FI}(X^{\mathrm{clone}}) = 1$. Thus, $\mathrm{FI}(X) = \frac{1}{2}$, and the FI of a variable can therefore be decreased if another variable is added. □

Relevance. It is important to observe that the FI of a variable is dependent on the other features (Properties 11 and 12). Adding or removing features could change the FI, which one needs to be aware of.

Property 13 (Cloning does not increase FI)**.** For any RV $X \in \Omega_{\mathrm{feat}}$, adding an identical variable $X^{\mathrm{clone}} := X$ (cloning) to the dataset, does not increase the FI of X.

Proof. Let $\mathrm{FI}_{\text{with clone}}(X)$ denote the FI of X after the clone X^{clone} is added. Using Equation (7.6) (with Notations (N1) and (N2)), we find

$$\begin{aligned}
\mathrm{FI}_{\text{with clone}}(X) &= \sum_{S \subseteq \Omega_{\mathrm{feat}} \cup \{X^{\mathrm{clone}}\} \setminus \{X\}} w(S, N_{\mathrm{vars}} + 1) \cdot D(X, Y, S) \\
&\overset{(a)}{=} \sum_{S \subseteq \Omega_{\mathrm{feat}} \cup \{X^{\mathrm{clone}}\} \setminus \{X\} : X^{\mathrm{clone}} \in S} w(S, N_{\mathrm{vars}} + 1) \cdot D(X, Y, S) \\
&\quad + \sum_{S \subseteq \Omega_{\mathrm{feat}} \cup \{X^{\mathrm{clone}}\} \setminus \{X\} : X^{\mathrm{clone}} \notin S} w(S, N_{\mathrm{vars}} + 1) \cdot D(X, Y, S) \\
&\overset{(b)}{=} \sum_{S \subseteq \Omega_{\mathrm{feat}} \cup \{X^{\mathrm{clone}}\} \setminus \{X\} : X^{\mathrm{clone}} \in S} w(S, N_{\mathrm{vars}} + 1) \cdot 0 \\
&\quad + \sum_{S \subseteq \Omega_{\mathrm{feat}} \cup \{X^{\mathrm{clone}}\} \setminus \{X\} : X^{\mathrm{clone}} \notin S} w(S, N_{\mathrm{vars}} + 1) \cdot D(X, Y, S) \\
&= \sum_{S \subseteq \Omega_{\mathrm{feat}} \setminus \{X\}} w(S, N_{\mathrm{vars}} + 1) \cdot D(X, Y, S).
\end{aligned}$$

Equality (a) follows by splitting the sum over all subsets of $\Omega_{\text{feat}} \cup \{X^{\text{clone}}\} \setminus \{X\}$ whether X^{clone} is part of the subset or not. Adding X to a subset that already contains the clone X^{clone} does not change the *BP* dependency function, which is why Equality (b) follows. The takeaway from this derivation is that the sum over all subsets $S \subseteq \Omega_{\text{feat}} \cup \{X^{\text{clone}}\} \setminus \{X\}$ reduces to the sum over $S \subseteq \Omega_{\text{feat}} \setminus \{X\}$.

Comparing the new $\text{FI}_{\text{with clone}}(X)$ with the original $\text{FI}(X)$ gives

$$\begin{aligned}\text{FI}(X) - \text{FI}_{\text{with clone}}(X) = &\sum_{S \subseteq \Omega_{\text{feat}} \setminus \{X\}} w(S, N_{\text{vars}}) \cdot D(X, Y, S) \\ &- \sum_{S \subseteq \Omega_{\text{feat}} \setminus \{X\}} w(S, N_{\text{vars}} + 1) \cdot D(X, Y, S).\end{aligned}$$

Using Notation (N1), we find that

$$\frac{w(S, N_{\text{vars}} + 1)}{w(S, N_{\text{vars}})} = \frac{\frac{|S|! \cdot (N_{\text{vars}} + 1 - |S| - 1)!}{(N_{\text{vars}} + 1)!}}{\frac{|S|! \cdot (N_{\text{vars}} - |S| - 1)!}{N_{\text{vars}}!}} = \frac{N_{\text{vars}} - |S|}{N_{\text{vars}} + 1} < 1,$$

thus $\text{FI}(X) - \text{FI}_{\text{with clone}}(X) \geq 0$ with equality if and only if $\text{FI}(X) = 0$. Therefore, we can conclude that cloning a variable cannot increase the FI of X and will decrease the FI when X is *not* null-independent. □

Relevance. We consider this a natural property of a good FI method, as no logical reason can be found why adding the exact same information would lead to an increase in FI for the original variable. The information a variable contains only becomes less valuable, as it becomes common knowledge.

Property 14 (Order does not change FI)**.** The order of the features does not affect the individually assigned FI. Consider the datasets $[X_1, X_2, \ldots, X_{N_{\text{vars}}}]$ and $[Z_1, Z_2, \ldots, Z_{N_{\text{vars}}}]$, where $Z_{\pi(i)} = X_i$ for some permutation π. It holds that $\text{FI}(X_i) = \text{FI}(Z_{\pi(i)})$ for any $i \in \{1, \ldots, N_{\text{vars}}\}$.

Proof. Note that the order of features nowhere plays a roll in the definition of BP-FI (Equation (7.6)). The *BP* dependency function is also independent of the given order, which is why this property trivially holds. □

Relevance. This is a very natural property of a good FI. Consider what would happen if the FI is dependent on the order in the dataset. Should all possible orders be evaluated and averaged to receive a final FI? We cannot find any arguments why someone should want FI to be dependent on the order of features.

Datasets

Next, we consider a few datasets, where we derive the theoretical outcome for the BP-FI. These datasets are also used in Section 7.4.3 to test FI methods. It is very hard to evaluate FI methods, as the ground truth is often unknown. However, we believe that the FI outcomes on these datasets are all natural and defendable. However, it remains subjective what one considers to be the 'correct' FI values.

Property 15 (XOR dataset)**.** Consider the following dataset consisting of two binary features $X_1, X_2 \sim \mathcal{U}(\{0,1\})$ and a target variable $Y = X_1 \cdot (1 - X_2) + X_2 \cdot (1 - X_1)$ which is the XOR of X_1 and X_2. It holds that

$$\mathrm{FI}(X_1) = \mathrm{FI}(X_2) = \frac{1}{2}.$$

Proof. Observe that $\mathrm{Dep}(Y|X_1) = \mathrm{Dep}(Y|X_2) = 0$ and $\mathrm{Dep}(Y|X_1 \cup X_2) = 1$. With Equation (7.6), it follows that

$$\begin{aligned}
\mathrm{FI}(X_1) &= \sum_{S \subseteq \{1,2\} \setminus X_1} \frac{|S|! \cdot (1 - |S|)!}{2!} \cdot (\mathrm{Dep}(Y|S \cup X_1) - \mathrm{Dep}(Y|S)) \\
&= \frac{|\{\varnothing\}|! \cdot (1 - |\{\varnothing\}|)!}{2!} \cdot (\mathrm{Dep}(Y|\{\varnothing\} \cup X_1) - \mathrm{Dep}(Y|\{\varnothing\})) \\
&\quad + \frac{|\{X_2\}|! \cdot (1 - |\{X_2\}|)!}{2!} \cdot (\mathrm{Dep}(Y|X_1 \cup X_2) - \mathrm{Dep}(Y|X_2)) \\
&= \frac{1}{2} \cdot (\mathrm{Dep}(Y|X_1) - 0) + \frac{1}{2} \cdot (\mathrm{Dep}(Y|X_1 \cup X_2) - \mathrm{Dep}(Y|X_2)) \\
&= \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot (1 - 0) \\
&= \frac{1}{2}.
\end{aligned}$$

Using Property 1, it follows that $\mathrm{FI}(X_2) = 1 - \mathrm{FI}(X_1) = \frac{1}{2}$. □

Relevance. This XOR formula is discussed and used to test FI methods in [55]. However, they only test for equality ($\mathrm{FI}(X_1) = \mathrm{FI}(X_2)$), not the specific value. Due to *symmetry*, we would also argue that both X_1 and

X_2 should get the same FI, as they fulfill the same role. Together, they fully determine Y, which is why the total FI should be one (see Property 6). Dividing this equally amongst the two variables, gives a logical desirable FI outcome of $\frac{1}{2}$ for each variable.

Property 16 (Probability dataset)**.** Consider the following dataset consisting of $Y = \lfloor X_S/2 \rfloor$ and $X_i = Z_i + (S-1)$ with $Z_i \sim \mathcal{U}(\{0,2\})$ for $i = 1,2$ and $\mathbb{P}(S=1) = p$, $\mathbb{P}(S=2) = 1-p$. It holds that

$$\mathrm{FI}(X_1) = p \text{ and } \mathrm{FI}(X_2) = 1-p.$$

Proof. Observe that by Equation (7.4)

$$\begin{aligned}
\mathrm{UD}(X_1, Y) &= \sum_{x_1 \in \{0,1,2,3\}} p_{X_1}(x_1) \cdot \sum_{y \in \{0,1\}} \left| p_{Y|X_1=x_1}(y) - p_Y(y) \right| \\
&= \sum_{x_1 \in \{0,2\}} p_{X_1}(x_1) \cdot \sum_{y \in \{0,1\}} \left| p_{Y|X_1=x_1}(y) - \frac{1}{2} \right| \\
&\qquad + \sum_{x_1 \in \{1,3\}} p_{X_1}(x_1) \cdot \sum_{y \in \{0,1\}} \left| p_{Y|X_1=x_1}(y) - \frac{1}{2} \right| \\
&= \sum_{x_1 \in \{0,2\}} \frac{p}{2} \cdot \left(\left| 1 - \frac{1}{2} \right| + \left| 0 - \frac{1}{2} \right| \right) \\
&\qquad + \sum_{x_1 \in \{1,3\}} \frac{1-p}{2} \cdot \sum_{y \in \{0,1\}} \left| p_Y(y) - p_Y(y) \right| \\
&= p.
\end{aligned}$$

Similarly, it follows that $\mathrm{UD}(X_2, Y) = 1-p$.

$$\begin{aligned}
\mathrm{UD}(Y, Y) &= \sum_{y' \in \{0,1\}} p_Y(y') \cdot \sum_{y \in \{0,1\}} \left| p_{Y|Y=y'}(y) - p_Y(y) \right| \\
&= \sum_{y' \in \{0,1\}} \frac{1}{2} \cdot \left(\left| 1 - \frac{1}{2} \right| + \left| 0 - \frac{1}{2} \right| \right) \\
&= 1.
\end{aligned}$$

From Equation (7.3), it follows that $\mathrm{Dep}(Y|X_1) = p$ and $\mathrm{Dep}(Y|X_2) = 1-p$. Additionally, note that knowing X_1 and X_2 fully determines Y, thus $\mathrm{Dep}(Y|X_1 \cup X_2) = 1$. With Equation (7.6), we now find

$$\begin{aligned}
\mathrm{FI}(X_1) &= \sum_{S \subseteq \{X_1,X_2\} \setminus X_1} \frac{|S|! \cdot (1-|S|)!}{2!} \cdot (\mathrm{Dep}(Y|S \cup X_1) - \mathrm{Dep}(Y|S)) \\
&= \frac{|\{\varnothing\}|! \cdot (1-|\{\varnothing\}|)!}{2!} \cdot (\mathrm{Dep}(Y|\{\varnothing\} \cup X_1) - \mathrm{Dep}(Y|\{\varnothing\})) \\
&\quad + \frac{|\{X_2\}|! \cdot (1-|\{X_2\}|)!}{2!} \cdot (\mathrm{Dep}(Y|X_1 \cup X_2) - \mathrm{Dep}(Y|X_2)) \\
&= \frac{1}{2} \cdot (\mathrm{Dep}(Y|X_1) - 0) + \frac{1}{2} \cdot (\mathrm{Dep}(Y|X_1 \cup X_2) - \mathrm{Dep}(Y|X_2)) \\
&= \frac{1}{2} \cdot (p - 0) + \frac{1}{2} \cdot (1 - (1-p)) \\
&= \frac{p}{2} + \frac{p}{2} = p.
\end{aligned}$$

Using Property 1, it follows that $\mathrm{FI}(X_2) = 1 - \mathrm{FI}(X_1) = 1 - p$. □

Relevance. At first glance, it is not immediately clear why these FI values are natural, which is why we discuss this dataset in more detail. S can be considered a selection parameter that determines if X_1 or X_2 is used for Y with probability p and $1-p$, respectively. X_i is constructed in such a way that it is uniformly drawn from $\{0,2\}$ or $\{1,3\}$ depending on S. However, as $Y = \lfloor X_S/2 \rfloor$, it holds that $X_S = 0$ and $X_S = 1$ give the same outcome for Y. The same holds for $X_S = 2$ and $X_S = 3$. Therefore, note that the distribution of Y is independent of the selection parameter S. Knowing X_1 gives the following information. First, S can be derived from the value of X_1. When $X_1 \in \{0,2\}$ it must hold that $S = 1$, and if $X_1 \in \{1,3\}$ it follows that $S = 2$. Second, when $S = 1$ it means that Y is fully determined by X_1. If $S = 2$, knowing that $X_1 = 1$ or $X_1 = 3$ does not provide any additional information about Y. With probability p knowing X_1 will fully determine Y, whereas with probability $1-p$, it will provide no information about the distribution of Y. The outcome $\mathrm{FI}(X_1) = p$, is therefore very natural. The same argumentation applies for X_2, which leads to $\mathrm{FI}(X_2) = 1-p$.

7.4 Comparing with existing methods

In the previous section, we showed that BP-FI has many desirable properties. Next, we evaluate for a large collection of FI methods if the properties hold for several synthetic datasets. Note that these datasets can only be used as counterexample, not as proof of a property. First, we discuss the in Section 7.4.1 the FI methods that are investigated. Second, we give the datasets (Section 7.4.2) and explain how they are used to test the properties (Section 7.4.3). The results are discussed in Section 7.4.4.

7.4.1 Alternative FI methods

A wide range of FI methods have been suggested for all kinds of situations. It is therefore first necessary to discuss the major categorical differences between them.

Global vs. local An important distinction to make for FI methods is whether they are constructed for *local* or *global* explanations. *Global* FI methods give an importance score for each feature over the entire dataset, whereas *local* FI methods explain which variables were important for a single example [59]. The global and local scores do not have to coincide: "features that are globally important may not be important in the local context, and vice versa" [144]. This research is focussed on global FI methods, but sometimes a local FI approach can be averaged out to obtain a global FI. For example, in [109] a local FI method is introduced called *Tree SHAP*. It is also used globally, by averaging the absolute values of the local FI.

Model-specific vs. model-agnostic A distinction within FI methods can be made between model-*specific* and -*agnostic* methods. *Model-specific* methods aim to find the FI using a prediction model such as a neural network or random forest, whereas *model-agnostic* methods do not use a prediction model. The BP-FI is model-agnostic, which therefore gives insights into the dataset. Whenever a model-specific method is used, the focus lies more on gaining information about the prediction model, not the dataset. In our tests, we use both model-specific and -agnostic methods.

Classification vs. regression Depending on the exact dataset, the target variable is either *categorical* or *numerical*, which is precisely the difference between *classification* and *regression*. Not all existing FI methods can handle both cases. In this research, we generate synthetic *classification* datasets, so we only examine FI methods that are intended for these cases. An additional

problem with regression datasets, is that continuous variables need to be converted to discrete bins. This conversion could drastically change the FI scores, which makes it harder to draw fair conclusions.

Collection We have gathered the largest known collection of FI methods from various sources [3, 8, 16, 31, 35, 38, 41, 55, 59, 67, 73, 95, 110, 118, 123, 128, 129, 139, 147, 148, 179, 180] or implemented them ourselves. This has been done with the following policy: Whenever code of a *classification* FI method was available in R or Python or the implementation was relatively straightforward, it was added to the collection. This resulted in 196 *base* methods and 468 total methods, as some base methods can be combined with *multiple* machine learning approaches or selection objectives, see Table 7.4.1. However, beware that most methods also contain additional parameters, which are not investigated in this research. The *default* values for these parameters are always used.

7.4.2 Synthetic datasets

Next, we briefly discuss the datasets that are used to test the properties described in Section 7.3 for alternative FI methods. In Section 7.A, we introduce each dataset and explain how they are generated. To draw fair conclusions, the datasets are not drawn randomly, but *fixed*. To give an example of how we do generate a dataset, we examine Dataset 1 *Binary system* (see Section 7.A), where the target variable Y is defined as $Y := \sum_{i=1}^{3} 2^{i-1} \cdot X_i$ with $X_i \sim \mathcal{U}(\{0,1\})$ for all $i \in \{1,2,3\}$. To get interpretable results, we draw each combination of X and Y values the *same number* of times. An example can be seen in Table 7.4.2. For most datasets, we draw 1,000 samples in total. However Datasets 6 and 7 consist of 2,000 samples to ensure null-independence. The datasets have been selected to be computationally inexpensive and to test many properties (see Section 7.4.3) with a limited number of datasets. An overview of the generated datasets can be found in Table 7.4.3 including the corresponding outcome of BP-FI. Section 7.A provides more technical details about the features and target variables.

7.4.3 Property evaluation

In Section 7.4.1, we gathered a collection of existing FI methods. In this section, we evaluate if these FI methods have the same desirable and proven properties of the BP-FI method (see Section 7.3). Due to the sheer number of FI methods (468), it is unfeasible to prove each property for every method.

Table 7.4.1: All evaluated FI methods: List of all FI methods that are evaluated in the experiments. The methods with superscript work in combination with multiple options: *Logistic Regression*[I, II, III], *Ridge*[I, II], *Linear Regression*[I, II], *Lasso*[I, II], *SGD Classifier*[I, III], *MLP Classifier*[I, II], *K Neighbors Classifier*[I, II], *Gradient Boosting Classifier*[I, II, IV], *AdaBoost Classifier*[I, II], *Gaussian NB*[I, II], *Bernoulli NB*[I, II], *Linear Discriminant Analysis*[I, II], *Decision Tree Classifier*[I, II, IV, V], *Random Forest Classifier*[I, II, IV, V], *SVC*[I], *CatBoost Classifier*[I, II], *LGBM Classifier*[I, II, IV], *XGB Classifier*[I, II, IV, VII], *XGBRF Classifier*[I, II, IV, VII], *ExtraTree Classifier*[IV, V], *ExtraTrees Classifier*[IV, V], *plsda*[VI], *splsda*[VI], *gini*[VIII], *entropy*[VIII], *NN1*[IX], *NN2*[IX]. This leads to a total of 468 FI methods from various sources [3, 8, 16, 31, 35, 38, 41, 55, 59, 67, 73, 95, 110, 118, 123, 128, 129, 139, 147, 148, 179, 180] or self-implemented.

Feature Importance Methods			
1. AdaBoost Classifier	2. Random Forest Classifier[VIII]	3. Extra Trees Classifier[VIII]	4. Gradient Boosting Classifier
5. SVR absolute weights	6. EL absolute weights	7. Permutation Importance Classifier[I]	8. PCA sum
9. PCA weighted	10. chi2	11. f classif	12. mutual info classif
13. KL divergence	14. R Mutual Information	15. Fisher Score	16. FeatureVec
17. R Varimp Classifier	18. R PIMP Classifier	19. Treeinterpreter Classifier[V]	20. DIFFI
21. Tree Classifier[IV]	22. Linear Classifier[III]	23. Permutation Classifier[I]	24. Partition Classifier[I]
25. Sampling Classifier[I]	26. Kernel Classifier[I]	27. Exact Classifier[I]	28. RFI Classifier[I]
29. CFI Classifier[I]	30. Sum Classifier[VI]	31. Weighted X Classifier[VI]	32. Weighted Y Classifier[VI]
33. f oneway	34. alexandergovern	35. pearsonr	36. spearmanr
37. pointbiserialr	38. kendalltau	39. weightedtau	40. somersd
41. linregress	42. siegelslopes	43. theilslopes	44. multiscale graphcorr
45. booster weight[VII]	46. booster gain[VII]	47. booster cover[VII]	48. snn
49. knn	50. bayesglm	51. lssvmRadial	52. rocc
53. ownn	54. ORFpls	55. rFerns	56. treebag
57. RRF	58. svmRadial	59. ctree2	60. evtree
61. pda	62. rpart	63. cforest	64. svmLinear
65. xyf	66. C5.0Tree	67. avNNet	68. kknn
69. svmRadialCost	70. gaussprRadial	71. FH.GBML	72. svmLinear2
73. bstSm	74. LogitBoost	75. wsrf	76. plr
77. xgbLinear	78. rf	79. null	80. protoclass
81. monmlp	82. Rborist	83. mlpWeightDecay	84. svmRadialWeights
85. mlpML	86. ctree	87. loclda	88. sdwd
89. mlpWeightDecayML	90. svmRadialSigma	91. bstTree	92. dnn
93. ordinalRF	94. pda2	95. BstLm	96. RRFglobal
97. mlp	98. rpart1SE	99. pcaNNet	100. ORFsvm
101. parRF	102. rpart2	103. gaussprPoly	104. C5.0Rules
105. rda	106. rbfDDA	107. multinom	108. gaussprLinear
109. svmPoly	110. knn	111. treebag	112. RRF
113. ctree2	114. evtree	115. pda	116. rpart
117. cforest	118. xyf	119. C5.0Tree	120. kknn
121. gaussprRadial	122. LogitBoost	123. wsrf	124. xgbLinear
125. rf	126. null	127. monmlp	128. Rborist
129. mlpWeightDecay	130. mlpML	131. ctree	132. mlpWeightDecayML
133. dnn	134. pda2	135. RRFglobal	136. mlp
137. rpart1SE	138. parRF	139. rpart2	140. gaussprPoly
141. C5.0Rules	142. rbfDDA	143. multinom	144. gaussprLinear
145. binaryConsistency	146. chiSquared	147. cramer	148. gainRatio
149. giniIndex	150. IEConsistency	151. IEPConsistency	152. mutualInformation
153. roughsetConsistency	154. ReliefFeatureSetMeasure	155. symmetricalUncertain	156. IteratedEstimator[II]
157. PermutationEstimator[II]	158. KernelEstimator[II]	159. SignEstimator[II]	160. Shapley[I]
161. Banzhaf	162. RF	163. Garson[IX]	164. VIANN[IX]
165. LOFO[IX]	166. Relief	167. ReliefF	168. RReliefF
169. fit criterion measure	170. f ratio measure	171. gini index	172. su measure
173. spearman corr	174. pearson corr	175. fechner corr	176. kendall corr
177. chi2 measure	178. anova	179. laplacian score	180. information gain
181. modified t score	182. MIM	183. MRMR	184. JMI
185. CIFE	186. CMIM	187. ICAP	188. DCSF
189. CFR	190. MRI	191. IWFS	192. NDFS
193. RFS	194. SPEC	195. MCFS	196. UDFS
197. R2	198. DC	199. BCDC	200. AIDC
201. HSIC	202. **BP-FI**		

Legend							
1-12	sklearn	[128]	13-20	Additional methods	[3, 31, 35, 59, 73, 118, 139, 148]	21-27	shap explainer	[110]
28-29	Relative feature importance	[16]	30-32	R vip	[67]	33-44	scipy stats	[180]
45-47	booster classifier	[38]	48-109	R caret classifier	[95]	110-144	R firm classifier	[67]
145-155	R FSinR Classifier	[8]	156-159	Sage Classifier	[41]	160-161	QII Averaged Classifier	[179]
162-165	Rebelosa Classifier	[147]	166-168	Relief Classifier	[123]	169-196	ITMO	[129]
197-201	Sunnies	[55]	202	**BP-FI**	-			

Table 7.4.2: Fixed draw: Example of how the datasets are drawn. Instead of drawing each possible outcome uniformly at random, we draw each combination an equal fixed number of times.

Outcome				# Drawn	
X_1	X_2	X_3	Y	Fixed	Uniform
0	0	0	0	125	133
0	0	1	4	125	129
0	1	0	2	125	121
0	1	1	6	125	109
1	0	0	1	125	136
1	0	1	5	125	124
1	1	0	3	125	115
1	1	1	7	125	133

Instead, we devise tests to find counterexamples of these properties using generated datasets (see Section 7.4.2). Due to the number of tests (18), we only discuss the parts that are not straightforward, as most test directly measure the corresponding property. An overview of each test can be found in Section 7.B. A summary of the tests can be found in Table 7.4.4, where it is outlined for each test which property is tested on which datasets.

Computational errors To allow for computational errors, we tolerate a margin of $\epsilon = 0.01$ in each test. If, e.g., an FI value should be zero, a score of 0.01 or -0.01 is still considered a *pass*, whereas an FI value of 0.05 is counted as a *fail*. Usually, this works in the favor of the FI method. However, in Test 9 we evaluate if the FI method assigns zero FI to variables that are not null-independent. In this case, we consider $|\text{FI}(X)| \leq \epsilon$ to be *zero*, as the datasets are constructed in such a way that variables are either null-independent or far from being null-independent.

Running time We limit the running time to one hour per dataset on an i7-12700K processor, whilst four algorithms are running simultaneously. The datasets consist of a small number of features with a very limited outcome space and the number of samples is either 1,000 or 2,000, which is why one hour is a reasonable amount of time.

NaN or infinite values In some cases, an FI method assigns NaN or $\pm\infty$ to a feature. How we handle these values depends on the test. E.g., we consider NaN to fall outside the range $[0, 1]$ (Tests 4 and 55), but when we evaluate if the sum of FI values remains stable (Test 2) or if two symmetric

Table 7.4.3: Overview of datasets: An overview of the generated datasets and the corresponding BP-FI outcome. The details of these datasets can be found in Section 7.A. They are used to evaluate if existing FI methods adhere to the same properties as BP-FI (see Section 7.4.3).

	Dataset	Variables	BP-FI outcome
Binary system	1. - base	(X_1, X_2, X_3)	$(0.333, 0.333, 0.333)$
	2. - clone	$(X_1^{\text{clone}}, X_1, X_2, X_3)$	$(0.202, 0.202, 0.298, 0.298)$
	3. - clone + 1x fully info.	$(X_1^{\text{clone}}, X_1, X_2, X_3, X_4^{\text{full}})$	$(0.148, 0.148, 0.183, 0.183, 0.338)$
	4. - clone + 2x fully info.	$(X_1^{\text{clone}}, X_1, X_2, X_3, X_4^{\text{full}}, X_5^{\text{full}})$	$(0.117, 0.117, 0.136, 0.136, 0.248, 0.248)$
	5. - clone + 2x fully info. (different order)	$(X_3, X_4^{\text{full}}, X_5^{\text{full}}, X_1^{\text{clone}}, X_1, X_2)$	$(0.136, 0.248, 0.248, 0.117, 0.117, 0.136)$
Null-independent system	6. - base	$(X_1^{\text{null-indep.}}, X_2^{\text{null-indep.}}, X_3^{\text{null-indep.}})$	$(0.000, 0.000, 0.000)$
	7. - constant variable	$(X_1^{\text{null-indep.}}, X_2^{\text{null-indep.}}, X_3^{\text{null-indep.}}, X_4^{\text{const, null-indep.}})$	$(0.000, 0.000, 0.000, 0.000)$
Increasing bins	8. - base	$(X_1^{\text{bins=10}}, X_2^{\text{bins=50}}, X_3^{\text{bins=1,000, full}})$	$(0.297, 0.342, 0.361)$
	9. - more variables	$(X_1^{\text{bins=10}}, X_2^{\text{bins=20}}, X_3^{\text{bins=50}}, X_4^{\text{bins=100}}, X_5^{\text{bins=1,000, full}})$	$(0.179, 0.193, 0.204, 0.208, 0.216)$
	10. - clone (different order)	$(X_3^{\text{bins=1,000, full}}, X_2^{\text{bins=50}}, X_1^{\text{bins=10}}, X_3^{\text{clone, full}})$	$(0.262, 0.253, 0.223, 0.262)$
Dependent system	11. - 1x fully info.	$(X_1^{\text{full}}, X_2^{\text{null-indep.}}, X_3^{\text{null-indep.}})$	$(1.000, 0.000, 0.000)$
	12. - 2x fully info.	$(X_1^{\text{full}}, X_2^{\text{full}}, X_3^{\text{null-indep.}})$	$(0.500, 0.500, 0.000)$
	13. - 3x fully info.	$(X_1^{\text{full}}, X_2^{\text{full}}, X_3^{\text{full}})$	$(0.333, 0.333, 0.333)$
XOR dataset	14. - base	(X_1, X_2)	$(0.500, 0.500)$
	15. - single variable	$(X_1^{\text{null-indep.}})$	(0.000)
	16. - clone	$(X_1^{\text{clone}}, X_1, X_2)$	$(0.167, 0.167, 0.667)$
	17. - null-independent	$(X_1, X_2, X_3^{\text{null-indep.}})$	$(0.500, 0.500, 0.000)$
Probability dataset	18-28. - for $p \in \{0, 0.1, \ldots, 1\}$	(X_1, X_2)	$(p, 1-p)$

Table 7.4.4: Overview of experiments: To evaluate if existing FI methods have the same properties as the BP-FI, we use the tests from Section 7.B on the datasets from Section 7.A. ✓means that the test is performed on this dataset. ↕(i) denotes that this dataset is used as baseline or in conjunction with dataset i. The details of the tests and datasets can be found in the appendix.

Test (Section 7.B)	Evaluates: Property/Corollary	Dataset (Section 7.A)																										
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2	1.1	↕(2-5)	✓	✓	✓	✓	↕(7)	✓	↕(9-10)	✓	✓	↕(12-13)	✓	✓	↕(16-17)		✓	✓												
3	2	✓	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓		✓	✓						✓					
4	3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
5	3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
6	4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
7	4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
8	5						✓	✓				✓	✓			✓		✓	✓										✓
9	5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
10	6											✓																	
11	8			✓	✓	✓			✓	✓	✓	✓	✓	✓					✓										✓
12	9								✓	✓	✓																		
13	11	↕(2)	✓↕(3)	✓↕(4)	✓		↕(7)	✓	↕(9-10)	✓	✓				✓↕(16-17)	↕(14)	✓	✓											
14	12	↕(2)	✓↕(3)	✓↕(4)	✓		↕(7)	✓	↕(9-10)	✓	✓				✓↕(16-17)	↕(14)	✓	✓											
15	13	↕(2)	✓						↕(10)		✓						↕(16)		✓										
16	14				↕(5)	✓													↕(28)	↕(27)	↕(26)	↕(25)	↕(24)		✓	✓	✓	✓	✓
17	15														✓			✓											
18	16																		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

features receive the same FI (Test 3), we consider twice NaN or twice $\pm\infty$ to be the same.

Property 9 (Limiting the outcome space) Property 9 states that applying any measurable function f to a RV X cannot increase the FI. In other words, $\text{FI}(X) \geq \text{FI}(f(X))$ holds. This property is tested using Datasets 8 to 10 (see Table 7.4.4). These datasets contain variables that are the outcome of binning the target variable using different number of bins. This is how Property 9 is tested, as it should hold that $\text{FI}(X_i) \geq \text{FI}(X_j)$, whenever X_i has more bins than X_j.

Properties 11 and 12 (Adding features can increase/decrease FI) In all other tests, the goal is to find a counterexample of the property. However, Tests 13 and 14 are designed to evaluate if a feature gets an increased/decreased FI when a feature is added. This increase/decrease should be more than ϵ. The datasets are chosen in such a way that both an increase and decrease could occur (according to the BP-FI). Only for these tests, we consider the test failed if no counterexample (increase/decrease) is found.

7.4.4 Evaluation results

An overview of the general results can be seen in Table 7.4.5, where the number of methods that *pass* and *fail* is given per test. Next, we highlight additional insights into the results of the experiments.

Best performing methods The top 20 FI methods that pass the most tests are given in Table 7.4.6. Out of 18 tests, the BP-FI passes all tests, which is as expected as we have proven in Section 7.3 that the BP-FI actually has these properties. Classifiers from *R FSinR Classifier* and *ITMO* fill 11 of the top 20 spots. Out of 11 R FSinR Classifier methods, six are in the top 20, which is quite remarkable. However, observe that the gap between the BP-FI method and the second best method is $18 - 11 = 7$ passed tests. Additionally, 424 out of 468 methods fail more than half of the tests. Figure 7.4.1 shows how frequently each number of passed tests occurs. A detailed overview of where each top 20 method fails, can be seen in Table 7.4.5. Note again that in Tests 13 and 14 it is considered a fail if adding features never increase or decrease the FI, respectively. It could be that these methods are in fact capable of increasing or decreasing, but for some reason do not with our datasets. Strikingly, most of these methods perform bad on the datasets with a desirable outcome (Tests 17 and 18).

Adding a variable without additional information (Test 2), also often leads to a change in total FI.

Table 7.4.5: Overview of the results: Each FI method is evaluated using the tests outlined in Section 7.B, which evaluates if the method adheres to the same properties as the BP-FI (see Section 7.3). This table summarizes out of 468 FI methods how many *pass* or *fail* the test. A distinction is made for the top 20 passing methods. Failing the test means that a counterexample is found. Note that passing the test does not 'prove' that the FI method actually has the property. *No result* indicates that the test could not be executed, because the running time of the FI method was too long or an error occurred.

	Test																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Overall																		
# Passed	1	92	45	438	200	97	132	283	97	31	141	241	243	314	365	172	13	5
# Failed	466	369	421	29	267	370	335	184	370	413	326	98	216	145	58	288	421	459
# No result	1	7	2	1	1	1	1	1	1	24	1	129	9	9	45	8	34	4
Top 20																		
# Passed	1	10	15	20	19	7	18	18	2	13	17	20	4	6	20	17	2	4
# Failed	19	10	5	0	1	13	2	2	18	7	3	0	16	14	0	3	17	16
# No result	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Test 1 In this test, it is evaluated if the sum of FI values is the same as the sum for BP-FI. At first, this seems a rather strict requirement. However, it holds for all datasets that were used that $\text{Dep}(Y|\Omega_{\text{feat}})$ is either zero or one. Thus, we essentially evaluate if the sum of FI is equal to one, when all variables collectively fully determine Y and zero if all variables are null-independent. The tests show that no FI method is able to pass this test, except for the BP-FI. To highlight some of the methods that came close: *162. Rebelosa Classifier RF*, *2. Random Forest Classifier entropy*, *2. Random Forest Classifier gini* only fail for the datasets where the sum should be zero (because of null-independence) and *1. AdaBoost Classifier* only does not pass on three of the four datasets based on the XOR function (see Section 7.A), where the sum should be one, but was zero instead. FI method *51. lssvmRadial* came closest with two fails. For the null-independent datasets (Datasets 6 and 7), it gives each feature an FI of 0.5, making the sum larger than zero.

Test 2 In Figure 7.4.2, a breakdown is given of where the sum of the FI values is unstable. The most errors are made with the *Binary system*

Table 7.4.6: Top 20: Out of 468 FI methods, these 20 methods pass the 18 tests given in Section 7.B the most often. These tests are designed to examine if an FI method adheres to the same properties as the BP-FI , given in Section 7.3. *Passed* means that the datasets from Section 7.A do not give a counterexample. Certainly, this does not mean that the FI method is proven to actually have this property. *Failed* means that a counterexample was found. *No result* indicates that the test could not be executed, because the running time of the FI method was too long or an error occurred.

		Combined result:		
Method		# Passed	# Failed	# No result
202.	**BP-FI**	**18**	**0**	**0**
147.	cramer	11	7	0
148.	gainRatio	11	7	0
153.	roughsetConsistency	11	7	0
155.	symmetricalUncertain	11	7	0
172.	su measure	11	7	0
88.	sdwd	10	7	1
3.	Extra Trees Classifier	10	8	0
116.	rpart	10	8	0
126.	null	10	8	0
145.	binaryConsistency	10	8	0
152.	mutualInformation	10	8	0
161.	Banzhaf Ridge	10	8	0
197.	R2	10	8	0
162.	RF	10	8	0
166.	Relief	10	8	0
173.	spearman corr	10	8	0
188.	DCSF	10	8	0
189.	CFR	10	8	0
191.	IWFS	10	8	0

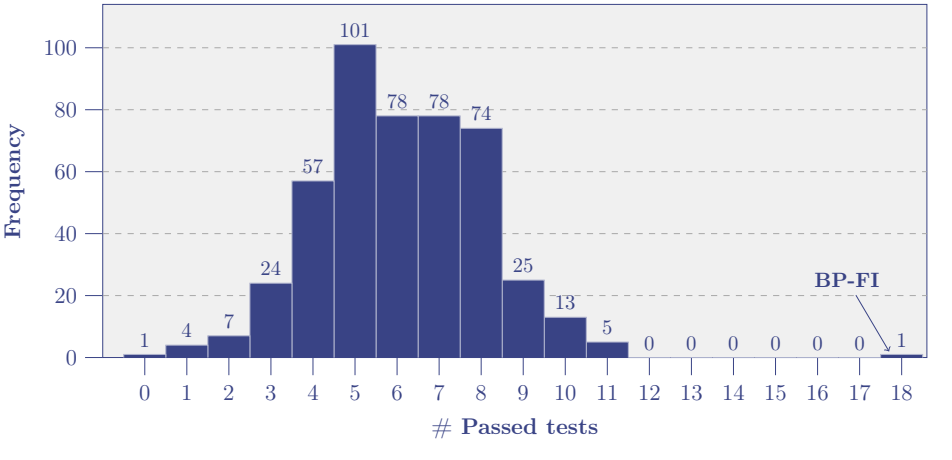


Figure 7.4.1: Frequency of total passed test: Histogram of the number of passed tests (out of 18) for the 468 FI methods.

datasets, when a fully informative feature is added. In total, 92 methods passed the test, whereas 369 failed. From these 369 methods, 279 fail with at least one increase of the sum, whereas 232 methods fail with at least one decrease. An alarming number of FI methods thus assign significantly more or less FI when a variable is added that does not contain any additional information. More or less credit is given out, whilst the collective knowledge is stable and does not warrant an increase or decrease in credit. Additionally, when the initial and final sum both contain a NaN value, it is considered as a pass. Three out of 92 would have not passed without this rule. If only the initial or the final sum contained NaN, it is considered a fail, because the sum is not the same. Only five methods fail solely by this rule: *15. Fisher Score*, *11. f classif*, *178. anova*, *179. laplacian score* and *192. NDFS*.

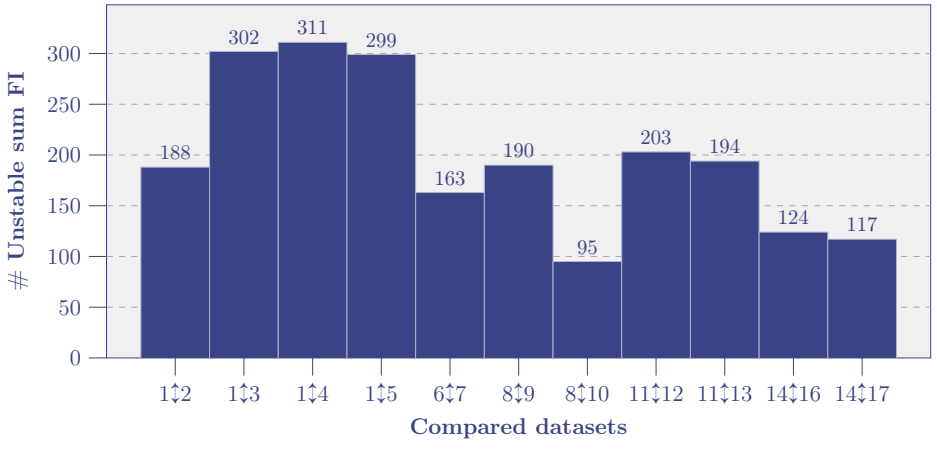


Figure 7.4.2: Unstable sum FI: Whenever a variable is added that does not give any additional information, the sum of all FI should remain stable. For each comparison, we determine how often this is not the case out of 468 FI methods.

Test 11 Figure 7.4.3 shows how often each variable is within an ϵ-bound of the largest FI in the dataset. Fully informative variables should attain the largest FI, according to Property 8. In total, we observe that the fully informative variables are often the largest FI with respect to the other variables. However, there still remain many cases where they are not. 326 FI methods fail this test, thus definitively not having Property 8. This makes interpretation difficult, when a variable can get more FI than a variable which fully determines the target variable. What does it mean, when a variable is more important than a variable that gives perfect information?

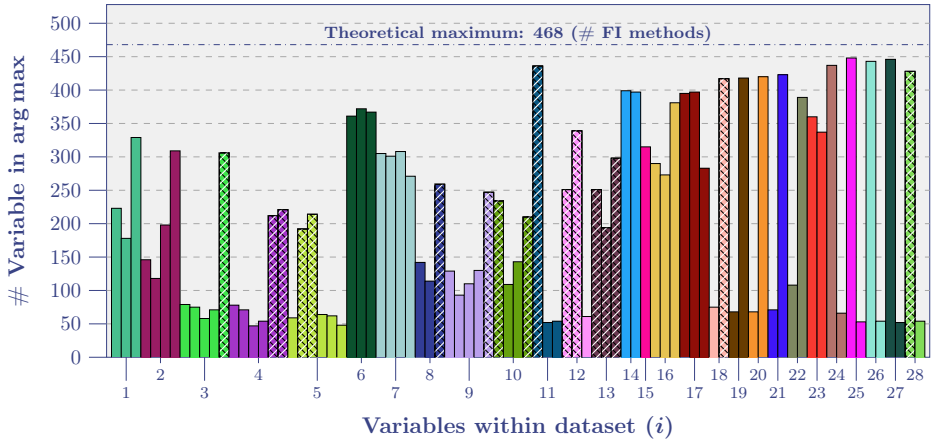


Figure 7.4.3: Argmax FI: For each variable in every dataset, we determine how often it receives the largest FI (within an ϵ-bound for $\epsilon = 0.01$) with respect to the other variables in the dataset. Fully informative variables should attain the largest FI (see Property 8). All fully informative variables are shaded in the figure.

Tests 10, 17, and 18 These tests all evaluate if the FI method assigns a specific value to a feature. From Table 7.4.5, we observe that not many methods are able to pass these tests. This is not surprising, as they have not been thoroughly tested yet to give a specific value. This is one of the important contributions of this research, which is why we want to elaborate on the attempts that have been made in previous research. A lot of synthetic datasets for FI have been proposed [2, 3, 6, 16, 30, 32, 44, 54, 55, 60, 71, 74, 80, 86, 104, 107–109, 117, 120, 124, 160, 164, 165, 169, 176, 187, 200, 202], but no specific desirable FI values were given. Most commonly, synthetic datasets are generated to evaluate the ability of an FI method to find *noisy* features [6, 30, 60, 71, 74, 80, 104, 160, 165, 169, 187, 200]. The common general concept of such a dataset is that the target variable is *independent* of certain variables. The FI values are commonly evaluated by comparing the FI values of independent variables with dependent variables with the goal to establish if the FI method is able to find independent variables. If the FI method actually predicts the exact desirable FI is not considered. Next, we highlight the papers where some comment about the desired FI is made. Lundberg et al. [109] give two similar datasets, where one variable *increases* in importance. They evaluate multiple FI methods to see if the same behavior is reflected in the outcome of these methods. This shows that some commonly used methods could assign lower importance to a variable,

when it should actually be increasing. Giles et al. [60] also design multiple artificial datasets to represent different scenarios, where comments are made about which variables should obtain more FI. Sundararajan et al. [169] remark that if every feature value is *unique*, that all variables get *equal* attributions for an FI method (CES) even if the function is not symmetric in the variables. If a tiny amount of noise is added to each feature, all features would get identical attributions. However, no assessment is done on the validity of this outcome. Owen et al. [124] give the following example. Let $f(x_1, x_2) = 10^6 x_1 + x_2$ with $x_1 = 10^6 x_2$, where they argue that, despite the larger variance of x_1, both variables are equally important, as the function can be written as a function of x_1 alone, but also only as a function of x_2. Although we have previously seen that 'written as a function of' is not a good criterion (due to dependencies), we agree with the authors that the FI should be equal. Another example is given by Owen et al. [124], where $\mathbb{P}(x_1 = 0, x_2 = 0, y = y_0) = p_0$, $\mathbb{P}(x_1 = 1, x_2 = 0, y = y_1) = p_1$, and $\mathbb{P}(x_1 = 0, x_2 = 1, y = y_2) = p_2$ are the possible outcomes. If $p_0 = 0$, it is stated in [124] that the Shapley relative importance of x_1 is $\frac{1}{2}$, which is "what it must be because there is then a bijection between x_1 and x_2". This is an interesting observation, as most papers do not comment about the validity of an outcome. Additionally, when $y_1 = y_2$ (and $y_0 \neq y_1$), Owen et al. [124] argue that the most important variable, is the one with the largest variance. Fryer et al. [55] also create a binary XOR dataset (see Dataset 14). They evaluate seven FI methods for this specific dataset. The role of X_1 and X_2 is symmetric, thus the assigned FI should also be identical. It is shown that six out of seven methods do indeed give a symmetrical result. However, the exact FI value varies greatly. *SHAP* gives FI of 3.19, whereas *Shapley DC* assigns 0.265 as FI. Only symmetry is checked, not the accuracy of the FI method. In conclusion, existing research was not focussed on predicting the exact accurate FI values. It is therefore not surprising that FI methods fail these accuracy tests so often. Table 7.4.7 outlines in more detail how often the variables are assigned an FI value outside an ϵ-bound (with $\epsilon = 0.01$) of the desired outcome. With Dataset 11, the FI methods mostly struggle with assigning 1 to the fully informative variable. In total, 413 methods failed Test 10. For Datasets 14 and 17, the two XOR variables fail about as often. Comparing these two datasets, it is interesting to note that the XOR variables fail more often, when a null-independent variable is added. In total, 421 methods failed Test 17. Test 18 is hard, as the FI method should assign the correct values for all probability datasets (see Section 7.A). Only five methods are able to pass this test: *152. mutualInformation*, *153. roughsetConsistency*, *162. RF*, *175. fechner corr*, and *202. BP-FI*. These five

methods also pass Test 10. However, besides BP-FI, there is only one method that also satisfies Test 17, which is *162. RF*. The other three methods all assign only zeros for Datasets 14 and 17, not identifying the value that the XOR variables hold, when their information is combined. In Figure 7.4.4, a breakdown is given for each probability dataset how often FI methods fail. An unexpected result, is that the dataset with probability $p < \frac{1}{2}$ and the dataset with probability $1 - p$ do not fail as often. Consistently, $p < \frac{1}{2}$ fails less often than its counterpart $1 - p$, although the datasets are the same up to a reordering of the features and the samples. This effect can also be seen in Table 7.4.7.

Table 7.4.7: Specific outcomes: Tests 10, 17, and 18 all evaluate if an FI method gives a specific outcome for certain dataset. In this table, it is outlined how often each variable of these datasets is assigned a value outside an ϵ-bound (with $\epsilon = 0.01$) of the desired outcome.

Dataset	Desirable outcome	# Non desirable outcome					
		not NaN			NaN		
		X_1	X_2	X_3	X_1	X_2	X_3
11	$(1, 0, 0)$	360	89	88	4	4	4
14	$(0.5, 0.5)$	353	351	-	5	5	-
17	$(0.5, 0.5, 0)$	369	364	90	5	5	5
18	$(0, 1)$	82	352	-	4	4	-
19	$(0.1, 0.9)$	412	434	-	3	3	-
20	$(0.2, 0.8)$	434	438	-	3	3	-
21	$(0.3, 0.7)$	435	441	-	3	3	-
22	$(0.4, 0.6)$	439	436	-	3	3	-
23	$(0.5, 0.5)$	423	422	-	3	3	-
24	$(0.6, 0.4)$	448	447	-	3	3	-
25	$(0.7, 0.3)$	449	446	-	3	3	-
26	$(0.8, 0.2)$	446	444	-	3	3	-
27	$(0.9, 0.1)$	444	435	-	3	3	-
28	$(1, 0)$	352	86	-	5	5	-

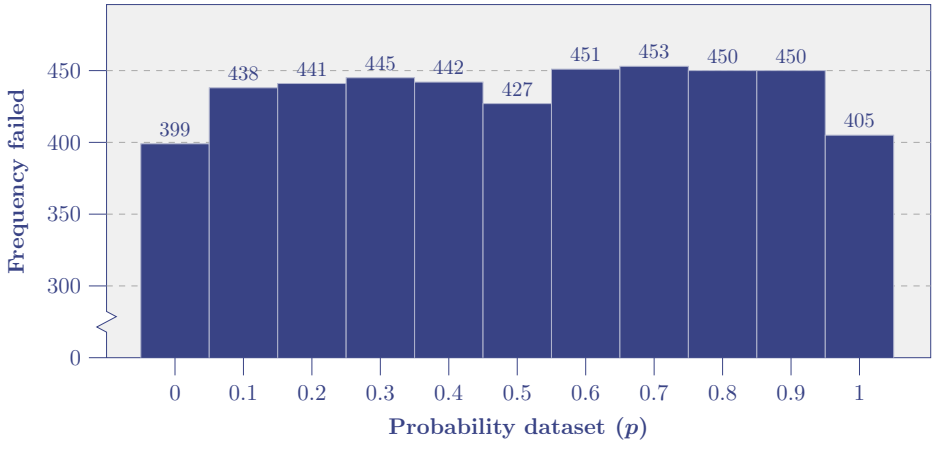


Figure 7.4.4: Breakdown Test 18 per dataset: In Test 18 an FI method needs to assign the correct FI values for every probability dataset (see Section 7.A). In this figure, we breakdown per dataset how often an FI method fails.

No result Focussing on the *no result* row of Table 7.4.5, there is one base method named *158. KernelEstimator* in combination with *Lasso* that in all cases did not work or exceeded running time. The large number of no results in Test 12 stem mostly from slow running times on the three datasets that are used in the test. At least 63 methods were too slow for each dataset, which automatically means that the test cannot be executed.

7.5 Discussion and future research

Whilst it is recommended to use our new FI method, it is important to understand the limitations and potential pitfalls. Below we elaborate on both the shortcomings of the approach proposed, and the related challenges for further research. We start by discussing by some matters that one needs to be aware of when applying the BP-FI (Section 7.5.1). Next, we discuss some choices that were made for the experiments in Section 7.5.2. Finally, we elaborate on other possible research avenues in Section 7.5.3.

7.5.1 Creating awareness

Binning Berkelmans et al. [13] explained that the way in which continuous data is discretized can have a considerable effect on the *BP* dependency function, which is why all datasets that were used in our research are *discrete*.

If a feature has too many unique values (due to poor binning), it will receive a higher FI from BP-FI, as more information can be stored in the unique values (see Property 9). On the other hand, when too few bins are chosen, an important feature can receive low FI, as the information is lost due to the binning. Future research should investigate and test which binning algorithms give the closest results to the underlying FI.

Too few samples Consider the following dataset: $X_i, Y \sim \mathcal{U}(\{0, 1, \ldots, 9\})$ i.i.d. for $i \in \{1, \ldots, 5\}$. Note that all features are null-independent, as Y is just uniformly drawn without considering the features in any way. If $n_{\text{samples}} = \infty$, the desired outcome would therefore be $(0, 0, 0, 0, 0)$. However, when *not enough* samples are given in the dataset, the features will get nonzero FI. Considering that the total number of different feature values is 10^5, combining all features *does* actually give information about Y, when $n_{\text{samples}} \ll 10^5$. For any possible combination of features, it is unlikely that it occurs more than once in the dataset. Therefore, knowing all feature values would (almost surely) determine the value of Y. Property 1 gives that the sum of all FI should therefore be one. All feature variables are also *symmetric* (Property 2), which is why the desired outcome is $(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5})$ instead. This example shows that one should be aware of the influence of the number of samples on the resulting FI. Variables that do *not* influence Y can still contain information, when not enough samples are provided. In this way, insufficient samples could lead to wrong conclusions, if one is not wary of this phenomenon.

Counterintuitive dependency case The *Berkelmans-Pries* dependency of Y on X measures how much probability mass of Y is shifted by knowing X. However, two similar shifts in probability mass could lead to different predictive power. To explain this, we examine the following dataset. $X_1, X_2 \sim \mathcal{U}(\{0, 1\})$ with

$$\mathbb{P}(Y = y | X_1 = x_1, X_2 = x_2) = \begin{cases} 1/4 & \text{if } (x_2, y) = (0, 0), \\ 3/4 & \text{if } (x_2, y) = (0, 1), \\ 5/8 & \text{if } (x_1, x_2, y) = (0, 1, 0), \\ 3/8 & \text{if } (x_1, x_2, y) = (0, 1, 1), \\ 7/8 & \text{if } (x_1, x_2, y) = (1, 1, 0), \\ 1/8 & \text{if } (x_1, x_2, y) = (1, 1, 1). \end{cases}$$

Knowing the value of X_2 shifts the distribution of Y. Before, Y was split 50/50, but when the value of X_2 is known, the labels are either split 25/75

or 75/25, depending on the value of X_2. Knowing X_1 gives even more information, as e.g., knowing $X_1 = X_2 = 1$ makes it more likely that $Y = 0$. However, the shift in distribution of Y is the same for knowing only X_2 and X_1 combined with X_2, which results in $\text{Dep}(Y|X_2) = \text{Dep}(Y|X_1 \cup X_2)$. This is a counterintuitive result. Globally, knowing X_2 or $X_1 \cup X_2$ gives the same shift in distribution, but locally we can predict Y much better if we know X_1 as well. We are unsure how this effects the BP-FI. In this case, it follows that $\text{FI}(X_1 \cup X_2) > \text{FI}(X_2)$, which is desirable. It is not unthinkable that a solution can be found to modify the dependency function in order to get a more intuitive result for such a case. Think e.g., of a different distance metric, that incorporates the local accuracy given the feature values or a conditional variant, which not only tests for independence, but also for conditional independence. These are all critical research paths that should be investigated.

Using FI for feature selection *Feature selection* (FS) is "the problem of choosing a small subset of features that ideally is necessary and sufficient to describe the target concept" [89]. Basically, the objective is to find a subset of all features that gives the best performance for a given model, as larger feature sets could decrease the accuracy of a model [97]. Many FI methods actually stem from a FS procedure. However, it is important to stress that *high* FI means that it should automatically be selected as feature. Shared knowledge with other features could render the feature less useful than expected. The other way around, *low* FI features should not automatically be discarded. In combination with other features, it could still give some additional insights that other features are not able to provide. Calculation of BP-FI values could also provide insight into which group of K features Y is most dependent on. To derive the result of BP-FI, all dependencies of Y on a subset $S \subseteq \Omega_{\text{feat}}$ are determined. If only K variables are selected, it is natural to choose

$$S_K^* \in \underset{S \subseteq \Omega_{\text{feat}}: |S| = K}{\arg\max} \{\text{Dep}(Y|S)\}.$$

These values are stored as an intermediate step in BP-FI, thus S_K^* can be derived quickly thereafter.

Larger outcome space leads to higher FI We have proven that a larger outcome space can never lead to a decrease in FI for BP-FI. This means, that features with more possible outcomes are more likely to attain a higher FI, depending on the distribution. There is a difference between

a feature that has many possible outcomes that are *almost never* attained, and a feature where many possible outcomes are *regularly* observed. We do not find this property undesirable, as some articles suggest [166, 200], as we would argue that a feature *can* contain more information by storing the information in additional outcomes, which would lead to an non-decreasing FI.

7.5.2 Experimental design choices

Regression To avoid binning issues, we only considered classification models and datasets. There are many more regression FI methods, that should be considered in a similar fashion. However, to draw clear and accurate conclusions, it is first necessary to understand how binning affects the results. Sometimes counterintuitive results can occur due to binning, that are not necessarily wrong. In such a case, it is crucial that the FI method is not depreciated.

Runtime In the experiments, it could happen that an FI method had *no result*, due to an excessive runtime or incompatible FI scores. The maximum runtime for each algorithm was set to one hour per dataset on an i7-12700K processor with 4 algorithms running simultaneously. The maximum runtime was necessary due to the sheer number of FI methods and datasets. Running four algorithms in parallel could unfairly penalize the runtime, as the processor is sometimes limited by other algorithms. In some occurrences, other parallel processes were already finished, which could potentially lower the runtime of an algorithm. There is a potential risk here, that accurate (but slow) FI methods are not showing up in the results. However, our synthetic datasets are relatively small with respect to the number of samples and the number of features, and we argue that one hour should be reasonable. Depending on the use case, sometimes a long time can be used to determine an FI value, whereas in other cases it could be essential to determine it rather quickly. Especially for larger datasets, it could even be unfeasible to run some FI methods. BP-FI uses Shapley values, which are exponentially harder to compute when the number of features grow. Approximation algorithms should be developed to faster estimate the true BP-FI outcome. Quick approximations could be useful if the runtime is much faster and the approximation is decent enough. Already, multiple papers have suggested approaches to approximate Shapley values faster [2, 33, 80, 105, 167]. These approaches save time, but at what cost? A study could be done to find the best FI method given a dataset and an allowed running time.

Stochasticity methods One factor we did not incorporate, is the *stochasticity* of some FI methods. Some methods do not predict the same FI values, when it is repeatedly used. As example, *79. rf* predicted for Dataset 3 (12.1, 11.7, 17.9, 15.2, 37.7) rounded to the first decimal. Running the method again gives a different result: (11.4, 12.0, 17.4, 15.6, 37.1), as this method uses a stochastic *random forest*. In principle, it is *undesirable* that an FI method is stochastic, as we believe that there should be a unique assignment of FI given a dataset. Due to the number of FI methods and datasets, we did not repeat and averaged each FI method. This would however give a better view on the performance of stochastic FI methods.

Parameter tuning All FI methods were used with *default* parameter values. Different parameter values could lead to more or less failed tests. However, the *ideal* parameter setting is not known beforehand, making it necessary to search a wide range of parameters. This was not the focus of our research, but future research could try to understand and learn which parameter values should be chosen for a given dataset.

Ranking FI methods In Table 7.4.6, the 20 FI methods that passed the most tests were highlighted. However, it is important to stress that not every test is equally difficult. Depending on the user, some properties could be more or less relevant. It is e.g., much harder to accurately predict the specific values for 11 datasets (Test 18), than to always predict non-negatively (Test 4). Every test is weighed equally, but this does not necessarily represent the difficulty of passing each test accurately. However, we note that *175. fechner corr* is the only FI method that passed Test 18, that ended up outside the top 20. We stress that we focussed on finding out if FI methods adhere to the properties, not necessarily finding the best and most fair ranking.

7.5.3 Additional matters

Global vs. local BP-FI is designed to determine the FI *globally*. However, another important research area focusses on *local* explanations. These explanations should provide information about why a specific sample has a certain target value instead of a different value. They provide the necessary interpretability that is increasingly demanded for practical applications. This could give insights for questions like: 'If my income would be higher, could I get a bigger loan?', 'Does race play a role in this prediction?', and 'For this automated machine learning decision, what were the critical factors?'. Many local FI methods have been proposed, and some even use Shapley values. A structured review should be made about all proposed local methods,

similar to our approach for global FI methods to find which local FI methods actually produce accurate explanations.

BP-FI can be modified to provide local explanations. For example, we can make the characteristic function localized in the following way. Let $Y_{S,z}$ be Y restricted to the event that $X_i = z_i$ for $i \notin S$, let us similarly define $X_{S,z}$. Then, we can define a localized characteristic function by:

$$v_z(S) := \text{Dep}\left(Y_{S,z} | X_{S,z}\right). \tag{7.8}$$

When dealing with continuous data, assuming equality could be too strict. In this case, a precision vector parameter ϵ can be used, where we define $Y_{S,z,\epsilon}$ to be Y restricted to the event that $|X_i - z_i| \leq \epsilon_i$ for $i \notin S$, and in the same way we define $X_{S,z,\epsilon}$. We then get the following localized characteristic function:

$$v_{z,\epsilon}(S) := \text{Dep}\left(Y_{S,z,\epsilon} | X_{S,z,\epsilon}\right).$$

Additionally, there are at least two possible ways how BP-FI can be adapted to be used for local explanations if some distance function $d(i, j)$ and parameter δ are available to determine if sample j is close enough to i to be considered 'local'. We can (I) discard all samples where $d(i, j) > \delta$ and/or (II) generate samples, such that $d(i, j) \leq \delta$ for all generated samples. Then, we can use BP-FI on the remaining samples and/or the generated samples, which would give local FI. Note that there should still be enough samples, as we have previously discussed that too few samples could lead to different FI outcomes. However, there are many more ways how BP-FI can be modified to be used for local explanations.

7

Model-specific FI BP-FI is in principle model-agnostic, as the FI is determined of the dataset, not the FI for a prediction model. However, BP-FI can still provide insights for any specific model. By replacing the target variable with the predicted outcomes of the model, we can apply BP-FI to this new dataset, which gives insight into which features are useful in the prediction model. Additionally, one can compare these FI results with the original FI (before replacing the target variable with the predicted outcomes) to see in what way the model changed the FI.

Additional properties In this research, we have proven properties of BP-FI. However, an in-depth study could lead to finding more useful properties. This holds both for BP-FI as well as the dependency function it is based

on. Applying isomorphisms e.g., does not change the dependency function. Therefore, the BP-FI is also stable under isomorphisms. Understanding what properties BP-FI has is a double-edged sword. Finding useful properties shows the power of BP-FI and finding undesirable behavior could lead to a future improvement.

Additional datasets Ground truths are often unknown for FI. In this research, we have given two kinds of datasets where the desirable outcomes are natural. It would however, be useful to create a *larger* collection of datasets both for global and local FI with an exact ground truth. We recognize that this could be a tall order, but we believe that it is essential to further improve FI methods.

Human labeling In some articles [110, 144], humans are used to evaluate explanations. An intriguing question to investigate is if humans are good at predicting FI. The BP-FI can be used as baseline to validate the values that are given by the participants. Are humans able to identify the correct order of FI? Even more difficult, can they predict close to the actual FI values?

7.6 Summary

We started by introducing a novel FI method named *Berkelmans-Pries* FI (BP-FI), which combines *Shapley* values and the *Berkelmans-Pries* dependency function [13]. In Section 7.3, we proved many useful properties of BP-FI. We discussed which FI methods already exist and introduced datasets to evaluate if these methods adhere to the same properties. In Section 7.4.3, we explain how the properties are tested. The results show that BP-FI is able to pass *many more* tests than any other FI method from a large collection of FI methods (468), which is a significant step forwards. Most methods have not previously been tested to give exact results due to missing ground truths. In this research, we provide several specific datasets, where the desired FI can be derived. From the tests, it follows that previous methods are not able to accurately predict the desired FI values. In Section 7.5, we extensively discussed the shortcomings of this chapter, and the challenges for further research. There are many challenging research opportunities that should be explored to further improve interpretability and explainability of datasets and machine learning models.

Appendix (Chapter 7)

7.A Datasets

In this appendix, we discuss how the datasets are generated that are used in the experiments. We use *fixed draw* instead of *uniformly random* to draw each dataset *exactly* according to its distribution. This is done to remove stochasticity from the dataset in order to get precise and interpretable results. An example of the difference between fixed draw and uniformly random can be seen in Table 7.4.2. The datasets consist of 1,000 samples, except for Datasets 6 and 7 which contains 2,000 samples to ensure null-independence. The datasets are designed to be computationally inexpensive, whilst still being able to test many properties (see Section 7.4.3). Below, we outline the formulas that are used to generate the datasets and give the corresponding FI values of our novel method BP-FI.

Dataset 1: Binary system
Feature variable(s): $X_i \sim \mathcal{U}(\{0,1\})$ i.i.d. for $i \in \{1,2,3\}$
Target variable: $Y := \sum_{i=1}^{3} 2^{i-1} \cdot X_i$.
Order: (X_1, X_2, X_3).
BP-FI: $(0.333, 0.333, 0.333)$.

Dataset 2: Binary system with clone
Feature variable(s): $X_i \sim \mathcal{U}(\{0,1\})$ i.i.d. for $i \in \{1,2,3\}$ and $X_1^{\text{clone}} := X_1$.
Target variable: $Y := \sum_{i=1}^{3} 2^{i-1} \cdot X_i$.
Order: $(X_1^{\text{clone}}, X_1, X_2, X_3)$.
BP-FI: $(0.202, 0.202, 0.298, 0.298)$.

Dataset 3: Binary system with clone and one fully informative variable
Feature variable(s): $X_i \sim \mathcal{U}(\{0,1\})$ i.i.d. for $i \in \{1,2,3\}$ and $X_1^{\text{clone}} := X_1$ and $X_4^{\text{full}} := Y^2$.
Target variable: $Y := \sum_{i=1}^{3} 2^{i-1} \cdot X_i$.
Order: $(X_1^{\text{clone}}, X_1, X_2, X_3, X_4^{\text{full}})$.
BP-FI: $(0.148, 0.148, 0.183, 0.183, 0.338)$.

Dataset 4: Binary system with clone and two fully informative variables
Feature variable(s): $X_i \sim \mathcal{U}(\{0,1\})$ i.i.d. for $i \in \{1,2,3\}$ and $X_1^{\text{clone}} := X_1$ and $X_4^{\text{full}} := Y^2$, $X_5^{\text{full}} := Y^3$.
Target variable: $Y := \sum_{i=1}^{3} 2^{i-1} \cdot X_i$.
Order: $(X_1^{\text{clone}}, X_1, X_2, X_3, X_4^{\text{full}}, X_5^{\text{full}})$.
BP-FI: $(0.117, 0.117, 0.136, 0.136, 0.248, 0.248)$.

Dataset 5: Binary system with clone and two fully informative variables different order
Feature variable(s): $X_i \sim \mathcal{U}(\{0,1\})$ i.i.d. for $i \in \{1,2,3\}$ and $X_1^{\text{clone}} := X_1$ and $X_4^{\text{full}} := Y^2$, $X_5^{\text{full}} := Y^3$.
Target variable: $Y := \sum_{i=1}^{3} 2^{i-1} \cdot X_i$.
Order: $(X_3, X_4^{\text{full}}, X_5^{\text{full}}, X_1^{\text{clone}}, X_1, X_2)$.
BP-FI: $(0.136, 0.248, 0.248, 0.117, 0.117, 0.136)$.

Dataset 6: Null-independent system
Feature variable(s): $X_i^{\text{null-indep.}} \sim \mathcal{U}(\{0,1\})$ i.i.d. for $i \in \{1,2,3\}$.
Target variable: $Y \sim \mathcal{U}(\{0,1\})$.
Order: $(X_1^{\text{null-indep.}}, X_2^{\text{null-indep.}}, X_3^{\text{null-indep.}})$.
BP-FI: $(0.000, 0.000, 0.000)$.

Dataset 7: Null-independent system with constant variable
Feature variable(s): $X_i^{\text{null-indep.}} \sim \mathcal{U}(\{0,1\})$ i.i.d. for $i \in \{1,2,3\}$ and $X_4^{\text{const, null-indep.}} := 1$.
Target variable: $Y \sim \mathcal{U}(\{0,1\})$.
Order: $(X_1^{\text{null-indep.}}, X_2^{\text{null-indep.}}, X_3^{\text{null-indep.}}, X_4^{\text{const, null-indep.}})$.
BP-FI: $(0.000, 0.000, 0.000, 0.000)$.

Dataset 8: Uniform system increasing bins
Feature variable(s): Let $\mathcal{L}_i := \{0, 1/(i-1), \ldots, 1\}$ be an equally spaced set. Define:

$$X_1^{\text{bins=10}} := \underset{x_1 \in \mathcal{L}_{10}}{\arg\max}\{Y \geq x_1\},$$

$$X_2^{\text{bins=50}} := \underset{x_2 \in \mathcal{L}_{50}}{\arg\max}\{Y \geq x_2\},$$

$$X_3^{\text{bins=1,000, full}} := \underset{x_3 \in \mathcal{L}_{1,000}}{\arg\max}\{Y \geq x_3\}.$$

Target variable: $Y \sim \mathcal{U}(\mathcal{L}_{1,000})$.
Order: $(X_1^{\text{bins=10}}, X_2^{\text{bins=50}}, X_3^{\text{bins=1,000, full}})$.
BP-FI: $(0.297, 0.342, 0.361)$.

Dataset 9: Uniform system increasing bins more variables
Feature variable(s): Let $\mathcal{L}_i := \{0, 1/(i-1), \ldots, 1\}$ be an equally spaced set. Define:

$$X_1^{\text{bins=10}} := \underset{x_1 \in \mathcal{L}_{10}}{\arg\max}\{Y \geq x_1\},$$

$$X_2^{\text{bins=20}} := \underset{x_2 \in \mathcal{L}_{20}}{\arg\max}\{Y \geq x_2\},$$

$$X_3^{\text{bins=50}} := \underset{x_3 \in \mathcal{L}_{50}}{\arg\max}\{Y \geq x_3\},$$

$$X_4^{\text{bins=100}} := \underset{x_4 \in \mathcal{L}_{100}}{\arg\max}\{Y \geq x_4\},$$

$$X_5^{\text{bins=1,000, full}} := \underset{x_5 \in \mathcal{L}_{1,000}}{\arg\max}\{Y \geq x_5\}.$$

Target variable: $Y \sim \mathcal{U}(\mathcal{L}_{1,000})$.
Order: $(X_1^{\text{bins=10}}, X_2^{\text{bins=20}}, X_3^{\text{bins=50}}, X_4^{\text{bins=100}}, X_5^{\text{bins=1,000, full}})$.
BP-FI: $(0.179, 0.193, 0.204, 0.208, 0.216)$.

Dataset 10: Uniform system increasing bins with clone different order
Feature variable(s): Let $\mathcal{L}_i := \{0, 1/(i-1), \ldots, 1\}$ be an equally spaced set. Define:

$$X_1^{\text{bins=10}} := \arg\max_{x_1 \in \mathcal{L}_{10}} \{Y \geq x_1\},$$

$$X_2^{\text{bins=50}} := \arg\max_{x_2 \in \mathcal{L}_{50}} \{Y \geq x_2\},$$

$$X_3^{\text{bins=1,000, full}} := \arg\max_{x_3 \in \mathcal{L}_{1,000}} \{Y \geq x_3\},$$

$$X_3^{\text{clone, full}} := X_3^{\text{bins=1,000, full}}.$$

Target variable: $Y \sim \mathcal{U}(\mathcal{L}_{1,000})$.
Order: $(X_3^{\text{bins=1,000, full}}, X_2^{\text{bins=50}}, X_1^{\text{bins=10}}, X_3^{\text{clone, full}})$.
BP-FI: $(0.262, 0.253, 0.223, 0.262)$.

Dataset 11: Dependent system: 1x fully informative variable
Feature variable(s): $X_1^{\text{full}}, X_2^{\text{null-indep.}}, X_3^{\text{null-indep.}} \sim \mathcal{U}(\{1, 2\})$.
Target variable: $Y := X_1^{\text{full}}$.
Order: $(X_1^{\text{full}}, X_2^{\text{null-indep.}}, X_3^{\text{null-indep.}})$.
BP-FI: $(1.000, 0.000, 0.000)$.

Dataset 12: Dependent system: 2x fully informative variable
Feature variable(s): $X_1^{\text{full}}, X_3^{\text{null-indep.}} \sim \mathcal{U}(\{1, 2\})$ and $X_2^{\text{full}} := Y^2$.
Target variable: $Y := X_1^{\text{full}}$.
Order: $(X_1^{\text{full}}, X_2^{\text{full}}, X_3^{\text{null-indep.}})$.
BP-FI: $(0.500, 0.500, 0.000)$.

Dataset 13: Dependent system: 3x fully informative variable
Feature variable(s): $X_1^{\text{full}} \sim \mathcal{U}(\{1, 2\})$ and $X_2^{\text{full}} := Y^2$, $X_3^{\text{full}} := Y^3$.
Target variable: $Y := X_1^{\text{full}}$.
Order: $(X_1^{\text{full}}, X_2^{\text{full}}, X_3^{\text{full}})$.
BP-FI: $(0.333, 0.333, 0.333)$.

Dataset 14: XOR dataset
Feature variable(s): $X_1, X_2 \sim \mathcal{U}(\{1, 2\})$.
Target variable: $Y := X_1 \cdot (1 - X_2) + X_2 \cdot (1 - X_1)$.
Order: (X_1, X_2).
BP-FI: $(0.500, 0.500)$.

Dataset 15: XOR dataset one variable
Feature variable(s): $X_1^{\text{null-indep.}} \sim \mathcal{U}(\{1,2\})$.
Target variable: $Y := X_1^{\text{null-indep.}} \cdot (1 - X_2) + X_2 \cdot (1 - X_1^{\text{null-indep.}})$ with $X_2 \sim \mathcal{U}(\{1,2\})$.
Order: $(X_1^{\text{null-indep.}})$.
BP-FI: (0.000).

Dataset 16: XOR dataset with clone
Feature variable(s): $X_1, X_2 \sim \mathcal{U}(\{1,2\})$ and $X_1^{\text{clone}} := X_1$.
Target variable: $Y := X_1 \cdot (1 - X_2) + X_2 \cdot (1 - X_1)$.
Order: $(X_1^{\text{clone}}, X_1, X_2)$.
BP-FI: $(0.167, 0.167, 0.667)$.

Dataset 17: XOR dataset with null independent
Feature variable(s): $X_1, X_2 \sim \mathcal{U}(\{1,2\})$ and $X_3^{\text{null-indep.}} \sim \mathcal{U}(\{0,3\})$.
Target variable: $Y := X_1 \cdot (1 - X_2) + X_2 \cdot (1 - X_1)$.
Order: $(X_1, X_2, X_3^{\text{null-indep.}})$.
BP-FI: $(0.500, 0.500, 0.000)$.

Dataset 18-28: Probability datasets
Feature variable(s): $X_i = Z_i + S$ with $Z_i \sim \mathcal{U}(\{0,2\})$ i.i.d. for $i = 1, 2$ and $\mathbb{P}(S = 1) = p$, $\mathbb{P}(S = 2) = 1 - p$.
Target variable: $Y = \lfloor X_S/2 \rfloor$.
Order: (X_1, X_2).
BP-FI: $(p, 1 - p)$.

7.B Tests

This appendix gives an overview of the tests that are used for each FI method to evaluate if they adhere to the properties given in Section 7.3. Most tests are straightforward, but additional explanations are given in Section 7.4.3.

Test 1: Efficiency sum BP-FI
Evaluates: Property 1.
Explanation: We evaluate if the sum of all FI is equal to the sum of the *Berkelmans-Pries* dependency function of Y on all features. When an FI value of NaN or infinite is assigned, the sum is automatically not equal to the sum for BP-FI.

Test 2: Efficiency stable
Evaluates: Corollary 1.1.
Explanation: Whenever a variable is added to a dataset, we examine if the sum of all FI changes. If a variable does not give any additional information compared to the other variables, the sum of all FI should stay the same.

Test 3: Symmetry
Evaluates: Property 2.
Explanation: In some datasets, there are *symmetrical* variables (see Property 2). We determine for all symmetrical variables if they receive identical FI.

Test 4: Range (lower)
Evaluates: Property 3.
Explanation: We examine for all FI outcomes if they are greater or equal to zero.

Test 5: Range (upper)
Evaluates: Property 3.
Explanation: We examine for all FI outcomes if they are smaller or equal to one.

Test 6: Bounds BP-FI (lower)
Evaluates: Property 4.
Explanation: We evaluate if the bounds given in Property 4 also hold for other FI methods. Every $\mathrm{FI}(X)$ with $X \in \Omega_{\text{feat}}$ can be lower bounded for BP-FI by $\frac{\mathrm{Dep}(Y|X)}{N_{\text{vars}}} \leq \mathrm{FI}(X)$.

Test 7: Bounds BP-FI (upper)
Evaluates: Property 4.
Explanation: We evaluate if the bounds given in Property 4 also hold for other FI methods. Every $\mathrm{FI}(X)$ with $X \in \Omega_{\text{feat}}$ can be upper bounded for BP-FI by $X \leq \mathrm{Dep}\left(Y|\Omega_{\text{feat}}\right)$.

Test 8: Null-independent implies zero FI
Evaluates: Property 5.
Explanation: In some datasets, there are *null-independent* variables. In these cases, we investigate if they also receive zero FI.

Test 9: Zero FI implies null-independent
Evaluates: Property 5.
Explanation: When a variable gets zero FI, it should hold that such a feature is null-independent.

Test 10: One fully informative, two null-independent
Evaluates: Property 6.
Explanation: feature importance: appendix: datasets) consists of a fully dependent target variable $Y := X_1^{\text{full}}$ and two null-independent variables $X_2^{\text{null-indep.}}, X_3^{\text{null-indep.}}$. We test if $\text{FI}(X_1^{\text{full}}) = 1$ and $\text{FI}(X_2^{\text{null-indep.}}) = \text{FI}(X_3^{\text{null-indep.}}) = 0$.

Test 11: Fully informative variable in argmax FI
Evaluates: Property 8.
Explanation: Whenever a fully informative feature exists in a dataset, there should not be a feature that attains a higher FI.

Test 12: Limiting the outcome space
Evaluates: Property 9.
Explanation: To evaluate if applying a measurable function f to a RV X could increase the FI, we examine the datasets where the same RV is binned using different bins. The binning can be viewed as applying a function f. Whenever less bins are used, the FI should not increase.

Test 13: Adding features can increase FI
Evaluates: Property 11.
Explanation: Whenever a feature is added to a dataset, we examine if this ever increases the FI of an original variable. If the FI never increases, we consider this a fail.

Test 14: Adding features can decrease FI
Evaluates: Property 12.
Explanation: Whenever a feature is added to a dataset, we examine if this ever decreases the FI of an original variable. If the FI never decreases, we consider this a fail.

Test 15: Cloning does not increase FI
Evaluates: Property 13.
Explanation: We evaluate if adding a clone to a dataset increase the FI of the original variable.

Test 16: Order does not change FI
Evaluates: Property 14.
Explanation: We check if the order of the variables changes the assigned FI.

Test 17: Outcome XOR
Evaluates: Property 15.
Explanation: This test evaluates the specific outcome of two datasets. For Dataset 14 the desired outcome is $(1/2, 1/2)$ and $(1/2, 1/2, 0)$ for Dataset 17. An FI method fails this test when one of the FI values falls outside the ϵ-bound of the desired outcome.

Test 18: Outcome probability datasets
Evaluates: Property 16.
Explanation: This test evaluates the specific outcomes of all probability datasets (Datasets 18 to 28). The desired outcome for probability p is $(p, 1-p)$. An FI method fails this test when one of the FI values falls outside the ϵ-bound of the desired outcome.

Bibliography

[1] R. de A. Araújo, A. L. Oliveira, and S. Meira. "A morphological neural network for binary classification problems". In: *Engineering Applications of Artificial Intelligence* 65 (Oct. 2017), pages 12–28. DOI: https://doi.org/10.1016/j.engappai.2017.07.014.

[2] K. Aas, M. Jullum, and A. Løland. "Explaining individual predictions when features are dependent: More accurate approximations to Shapley values". In: *Artificial Intelligence* 298 (2021), page 103502. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2021.103502. URL: https://www.sciencedirect.com/science/article/pii/S0004370221000539.

[3] N. Abe and M. Kudo. "Entropy criterion for classifier-independent feature selection". In: *Knowledge-Based Intelligent Information and Engineering Systems*. Edited by R. Khosla, R. J. Howlett, and L. C. Jain. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pages 689–695. ISBN: 978-3-540-31997-9.

[4] R. Agarwal, P. Sacre, and S. V. Sarma. *Mutual dependence: A novel method for computing dependencies between random variables*. 2015. arXiv: 1506.00673.

[5] C. Aggarwal, X. Kong, Q. Gu, J. Han, and P. Yu. "Active learning: A survey". English (US). In: *Data Classification*. Edited by C. Aggarwal. Publisher Copyright: © 2015 by Taylor & Francis Group, LLC.

CRC Press, Jan. 2014, pages 571–605. ISBN: 9781466586741. DOI: 10.1201/b17320.

[6] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer. "Permutation importance: A corrected feature importance measure". In: *Bioinformatics* 26.10 (Apr. 2010), pages 1340–1347. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btq134. eprint: https://academic.oup.com/bioinformatics/article-pdf/26/10/1340/16892402/btq134.pdf. URL: https://doi.org/10.1093/bioinformatics/btq134.

[7] B. Amos, B. Ludwiczuk, and M. Satyanarayanan. *OpenFace: A general-purpose face recognition library with mobile applications*. Technical report. CMU-CS-16-118, CMU School of Computer Science, 2016.

[8] F. Aragón-Royón, A. Jiménez-Vílchez, A. Arauzo-Azofra, and J. M. Benítez. *FSinR: An exhaustive package for feature selection*. 2020. DOI: 10.48550/ARXIV.2002.10330. URL: https://arxiv.org/abs/2002.10330.

[9] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. "Generalization and equilibrium in generative adversarial nets (GANs)". In: *CoRR* abs/1703.00573 (2017). arXiv: 1703.00573. URL: http://arxiv.org/abs/1703.00573.

[10] M. Artin. *Algebra*. 2nd. Pearson Education, 2011.

[11] J. Balayla. "Prevalence threshold (ϕe) and the geometry of screening curves". In: *PLoS ONE* 15.10 (Oct. 2020). Edited by A. D. Hutson, e0240215. DOI: https://doi.org/10.1371/journal.pone.0240215.

[12] G. Berkelmans. "Turing and Van Gogh walk into a bar". PhD thesis. Vrije Universiteit, 2023.

[13] G. Berkelmans, S. Bhulai, R. van der mei, and J. Pries. "The Berkelmans-Pries dependency function: A generic measure of dependence between random variables". In: *Journal of Applied Probability* (2023), pages 1–21. DOI: 10.1017/jpr.2022.118.

[14] E. van de Bijl, J. Klein, J. Pries, S. Bhulai, M. Hoogendoorn, and R. van der Mei. *The Dutch Draw: Constructing a universal baseline for binary prediction models*. 2022. DOI: 10.48550/ARXIV.2203.13084. URL: https://arxiv.org/abs/2203.13084.

[15] E. van de Bijl, J. Klein, J. Pries, R. van der Mei, and S. Bhulai. "Detecting novel application layer cybervariants using supervised learning". In: *International Journal On Advances in Security* 15.3 & 4 (2022), pages 75 –85. ISSN: 1942-2636. URL: http://www.iariajournals.org/security/.

[16] A. del Bimbo, R. Cucchiara, S. Sclaroff, G. M. Farinella, T. Mei, M. Bertini, H. J. Escalante, and R. Vezzani, editors. *Pattern Recognition. ICPR International Workshops and Challenges*. Springer International Publishing, 2021. DOI: 10.1007/978-3-030-68787-8. URL: https://doi.org/10.1007%2F978-3-030-68787-8.

[17] J.-C. de Borda. "Mémoire sur les élections au scrutin". In: *Histoire de l'Académie Royale des Sciences* (1781).

[18] A. Borji. *Pros and cons of GAN evaluation measures*. 2018. DOI: 10.48550/ARXIV.1802.03446. URL: https://arxiv.org/abs/1802.03446.

[19] A. Borji. *Pros and cons of GAN evaluation measures: New developments*. 2021. DOI: 10.48550/ARXIV.2103.09396. URL: https://arxiv.org/abs/2103.09396.

[20] R. C. Bradley. "Basic properties of strong mixing conditions. A survey and some open questions". In: *Probability Surveys* 2.none (2005), pages 107 –144. DOI: 10.1214/154957805100000104. URL: https://doi.org/10.1214/154957805100000104.

[21] G. Bradski. "The OpenCV library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[22] M. D. Breitenstein, D. Kuettel, T. Weise, L. van Gool, and H. Pfister. "Real-time face pose estimation from single range images". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pages 1–8.

[23] A. Brock, J. Donahue, and K. Simonyan. *Large scale GAN training for high fidelity natural image synthesis*. 2018. arXiv: 1809.11096.

[24] C. Buckley, M. O'Reilly, D. Whelan, A. V. Farrell, L. Clark, V. Longo, M. Gilchrist, and B. Caulfield. "Binary classification of running fatigue using a single inertial measurement unit". In: *2017 IEEE 14th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. 2017, pages 197–201. DOI: 10.1109/BSN.2017.7936040.

[25] J. Buolamwini and T. Gebru. "Gender shades: Intersectional accuracy disparities in commercial gender classification". In: *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*. Edited by S. A. Friedler and C. Wilson. Volume 81. Proceedings of Machine Learning Research. New York, NY, USA: PMLR, Feb. 2018, pages 77–91. URL: http://proceedings.mlr.press/v81/buolamwini18a.html.

[26] M. W. Callaghan and F. Müller-Hansen. "Statistical stopping criteria for automated screening in systematic reviews". In: *Systematic Reviews* 9.1 (Nov. 2020), page 273. ISSN: 2046-4053. DOI: 10.1186/

s13643-020-01521-4. URL: https://doi.org/10.1186/s13643-020-01521-4.

[27] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander. "Hierarchical density estimates for data clustering, visualization, and outlier detection". In: *ACM Transactions on Knowledge Discovery from Data* 10.1 (July 2015). ISSN: 1556-4681. DOI: 10.1145/2733381. URL: https://doi.org/10.1145/2733381.

[28] G. Canbek, S. Sagiroglu, T. T. Temizel, and N. Baykal. "Binary classification performance measures/metrics: A comprehensive visualized roadmap to gain new insights". In: *2017 International Conference on Computer Science and Engineering (UBMK)*. IEEE, Oct. 2017. DOI: https://doi.org/10.1109/ubmk.2017.8093539.

[29] L. Capitani, L. Bagnato, and A. Punzo. "Testing serial independence via density-based measures of divergence". In: *Methodology And Computing In Applied Probability* 16 (Aug. 2014), pages 627–641. DOI: 10.1007/s11009-013-9320-4.

[30] M. Carletti, C. Masiero, A. Beghi, and G. A. Susto. "Explainable machine learning in industry 4.0: Evaluating feature importance in anomaly detection to enable root cause analysis". In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. 2019, pages 21–26. DOI: 10.1109/SMC.2019.8913901.

[31] M. Carletti, M. Terzi, and G. A. Susto. *Interpretable anomaly detection with DIFFI: Depth-based isolation forest feature importance*. 2020. DOI: 10.48550/ARXIV.2007.11117. URL: https://arxiv.org/abs/2007.11117.

[32] G. Casalicchio, C. Molnar, and B. Bischl. "Visualizing the feature importance for black box models". In: *Machine Learning and Knowledge Discovery in Databases*. Edited by M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley, and G. Ifrim. Cham: Springer International Publishing, 2019, pages 655–670. ISBN: 978-3-030-10925-7.

[33] J. Castro, D. Gómez, and J. Tejada. "Polynomial calculation of the Shapley value based on sampling". In: *Computers & Operations Research* 36.5 (2009). Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X), pages 1726–1730. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2008.04.004. URL: https://www.sciencedirect.com/science/article/pii/S0305054808000804.

[34] A.-L. Cauchy. "Mémoire sur le nombre des valeurs qu'une fonction peut acquérir lorsqu'on y permute de toutes les maniéres possibles

les quantités qu'elle Renferme". In: *Journal de l'École polytechnique* (1815).

[35] E. Celik. *vita: Variable importance testing approaches*. R package version 1.0.0. 2015. URL: https://CRAN.R-project.org/package=vita.

[36] H. Chang and D.-Y. Yeung. "Robust path-based spectral clustering". In: *Pattern Recognition* 41.1 (2008), pages 191–203. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2007.04.010. URL: https://www.sciencedirect.com/science/article/pii/S0031320307002038.

[37] O. Chapelle, B. Schlkopf, and A. Zien. "Semi-supervised learning". In: *IEEE Transactions on Neural Networks* 20 (2006).

[38] T. Chen and C. Guestrin. "XGBoost: A scalable tree boosting system". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pages 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785.

[39] N. Chinchor. "MUC-4 evaluation metrics". In: *Proceedings of the 4th Conference on Message Understanding*. MUC4 '92. McLean, Virginia: Association for Computational Linguistics, 1992, pages 22–29. ISBN: 1558602739. DOI: https://doi.org/10.3115/1072064.1072067.

[40] R. Couronné, P. Probst, and A.-L. Boulesteix. "Random forest versus logistic regression: A large-scale benchmark experiment". In: *BMC Bioinformatics* 19.1 (July 2018). DOI: https://doi.org/10.1186/s12859-018-2264-5.

[41] I. Covert, S. M. Lundberg, and S.-I. Lee. "Understanding global feature contributions with additive importance measures". In: *Advances in Neural Information Processing Systems*. Edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Volume 33. Curran Associates, Inc., 2020, pages 17212–17223. URL: https://proceedings.neurips.cc/paper/2020/file/c7bf0b7c1a86d5eb3be2c722cf2cf746-Paper.pdf.

[42] G. Dasarathy, R. Nowak, and X. Zhu. "S2: An efficient graph based active learning algorithm with application to nonparametric classification". In: *Proceedings of The 28th Conference on Learning Theory*. Edited by P. Grünwald, E. Hazan, and S. Kale. Volume 40. Proceedings of Machine Learning Research. Paris, France: PMLR, July 2015, pages 503–522. URL: https://proceedings.mlr.press/v40/Dasarathy15.html.

[43] A. Datta, S. Sen, and Y. Zick. "Algorithmic transparency via Quantitative Input Influence: Theory and experiments with learning systems". In: *2016 IEEE Symposium on Security and Privacy (SP)*. 2016, pages 598–617. DOI: 10.1109/SP.2016.42.

[44] K. Dhamdhere, A. Agarwal, and M. Sundararajan. "The Shapley Taylor interaction index". In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org, 2020.

[45] J. M. Díaz Barros, B. Mirbach, F. Garcia, K. Varanasi, and D. Stricker. "Real-time head pose estimation by tracking and detection of keypoints and facial landmarks". In: *Computer Vision, Imaging and Computer Graphics Theory and Applications*. Edited by D. Bechmann, M. Chessa, A. P. Cláudio, F. Imai, A. Kerren, P. Richard, A. Telea, and A. Tremeau. Cham: Springer International Publishing, 2019, pages 326–349. ISBN: 978-3-030-26756-8.

[46] J. D. Dixon and B. Mortimer. *Permutation Groups*. Volume 163. Springer Science & Business Media, 1996. DOI: https://doi.org/10.1007/978-1-4612-0731-3.

[47] S. Ebert, M. Fritz, and B. Schiele. "Active metric learning for object recognition". In: *Pattern Recognition*. Edited by A. Pinz, T. Pock, H. Bischof, and F. Leberl. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pages 327–336. ISBN: 978-3-642-32717-9.

[48] P. Embrechts, A. J. McNeil, and D. Straumann. "Correlation and dependence in risk management: Properties and pitfalls". In: *Risk Management: Value at Risk and Beyond*. Cambridge: Cambridge University Press, 2002, pages 176–223. DOI: 10.1017/CBO9780511615337.008.

[49] B. Eriksson, G. Dasarathy, A. Singh, and R. Nowak. "Active clustering: Robust and efficient hierarchical clustering using adaptively selected similarities". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Edited by G. Gordon, D. Dunson, and M. Dudík. Volume 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Apr. 2011, pages 260–268. URL: https://proceedings.mlr.press/v15/eriksson11a.html.

[50] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "A density-based algorithm for discovering clusters in large spatial databases with noise". In: AAAI Press, 1996, pages 226–231.

[51] E. B. Fowlkes and C. L. Mallows. "A method for comparing two hierarchical clusterings". In: *Journal of the American Statistical Association* 78.383 (Sept. 1983), pages 553–569. DOI: https://doi.org/10.1080/01621459.1983.10478008.

[52] P. Fränti and O. Virmajoki. "Iterative shrinking method for clustering problems". In: *Pattern Recognition* 39.5 (2006), pages 761–765. DOI: 10.1016/j.patcog.2005.09.012. URL: http://dx.doi.org/10.1016/j.patcog.2005.09.012.

[53] P. Fränti and S. Sieranoja. *K-means properties on six clustering benchmark datasets*. 2018. URL: http://cs.uef.fi/sipu/datasets/.

[54] C. Frye, C. Rowat, and I. Feige. "Asymmetric Shapley values: Incorporating causal knowledge into model-agnostic explainability". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS'20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[55] D. V. Fryer, I. Strumke, and H. Nguyen. "Model independent feature attributions: Shapley values that uncover non-linear dependencies". In: *PeerJ Computer Science* 7 (2021), e582.

[56] L. Fu and E. Medico. "FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data". In: *BMC Bioinformatics* 8.1 (Jan. 2007), page 3. ISSN: 1471-2105. DOI: 10.1186/1471-2105-8-3. URL: https://doi.org/10.1186/1471-2105-8-3.

[57] Y. Gal, R. Islam, and Z. Ghahramani. "Deep Bayesian active learning with image data". In: *Proceedings of the 34th International Conference on Machine Learning*. Edited by D. Precup and Y. W. Teh. Volume 70. Proceedings of Machine Learning Research. PMLR, Aug. 2017, pages 1183–1192. URL: https://proceedings.mlr.press/v70/gal17a.html.

[58] H. Gebelein. "Das statistische problem der korrelation als variations- und eigenwertproblem und sein zusammenhang mit der ausgleichsrechnung". In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 21 (6 Jan. 1941), pages 364–379. ISSN: 1521-4001. DOI: 10.1002/ZAMM.19410210604. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/zamm.19410210604.

[59] A. Ghorbani, D. Berenbaum, M. Ivgi, Y. Dafna, and J. Y. Zou. "Beyond importance scores: Interpreting tabular ml by visualizing feature semantics". In: *Information* 13.1 (2022). ISSN: 2078-2489. DOI: 10.3390/info13010015. URL: https://www.mdpi.com/2078-2489/13/1/15.

[60] O. Giles et al. *Faking feature importance: A cautionary tale on the use of differentially-private synthetic data*. 2022. DOI: 10.48550/ARXIV.2203.01363. URL: https://arxiv.org/abs/2203.01363.

[61] A. Gionis, H. Mannila, and P. Tsaparas. "Clustering aggregation". In: *ACM Transactions on Knowledge Discovery from Data* 1.1 (Mar. 2007), 4–es. ISSN: 1556-4681. DOI: 10.1145/1217299.1217303. URL: https://doi.org/10.1145/1217299.1217303.

[62] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[63] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial nets". In: *Advances in Neural Information Processing Systems 27*. Edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pages 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[64] A. Gramacki. *Nonparametric Kernel Density Estimation and Its Computational Aspects*. 1st. New York: Springer Publishing Company, Incorporated, 2017. ISBN: 3319716875.

[65] M. Grandini, E. Bagli, and G. Visani. *Metrics for multi-class classification: An overview*. 2020. arXiv: 2008.05756.

[66] C. W. Granger, E. Maasoumi, and J. Racine. "A dependence metric for possibly nonlinear processes". In: *Journal of Time Series Analysis* 25.5 (2004), pages 649–669. DOI: https://doi.org/10.1111/j.1467-9892.2004.01866.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9892.2004.01866.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9892.2004.01866.x.

[67] B. M. Greenwell and B. C. Boehmke. "Variable importance plots—An introduction to the vip package". In: *The R Journal* 12.1 (2020), pages 343–366. URL: https://doi.org/10.32614/RJ-2020-013.

[68] A. Gretton, R. Herbrich, A. Smola, O. Bousquet, and B. Schölkopf. "Kernel methods for measuring independence". In: *Journal of Machine Learning Research* 6.70 (2005), pages 2075–2129. URL: http://jmlr.org/papers/v6/gretton05a.html.

[69] E. Hellinger. "Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen." In: *Journal für die reine und angewandte Mathematik* 1909.136 (1909), pages 210–271. DOI: doi:10.1515/crll.1909.136.210. URL: https://doi.org/10.1515/crll.1909.136.210.

[70] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter. "GANs trained by a two time-scale update rule converge to a Nash equilibrium". In: *CoRR* abs/1706.08500 (2017). arXiv: 1706.08500. URL: http://arxiv.org/abs/1706.08500.

[71] S. Hooker, D. Erhan, P.-J. Kindermans, and B. Kim. "A benchmark for interpretability methods in deep neural networks". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[72] H. Hotelling. "Relations between two sets of variates". In: *Biometrika* 28.3/4 (1936), pages 321–377. ISSN: 00063444. URL: http://www.jstor.org/stable/2333955 (visited on 06/02/2022).

[73] T. Hothorn and A. Zeileis. "partykit: A modular toolkit for recursive partytioning in R". In: *Journal of Machine Learning Research* 16 (2015), pages 3905–3909. URL: https://jmlr.org/papers/v16/hothorn15a.html.

[74] V. A. Huynh-Thu, Y. Saeys, L. Wehenkel, and P. Geurts. "Statistical interpretation of machine learning-based feature importance scores for biomarker discovery". en. In: *Bioinformatics* 28.13 (Apr. 2012), pages 1766–1774.

[75] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. "Label ranking by learning pairwise preferences". In: *Artificial Intelligence* 172.16 (2008), pages 1897–1916. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2008.08.002. URL: https://www.sciencedirect.com/science/article/pii/S000437020800101X.

[76] H. Ishibashi and H. Hino. "Stopping criterion for active learning based on deterministic generalization bounds". In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Edited by S. Chiappa and R. Calandra. Volume 108. Proceedings of Machine Learning Research. PMLR, Aug. 2020, pages 386–397. URL: https://proceedings.mlr.press/v108/ishibashi20a.html.

[77] A. K. Jain and M. H. C. Law. "Data clustering: A user's dilemma". In: *Pattern Recognition and Machine Intelligence*. Edited by S. K. Pal, S. Bandyopadhyay, and S. Biswas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pages 1–10. ISBN: 978-3-540-32420-1.

[78] R. J. Janse, T. Hoekstra, K. J. Jager, C. Zoccali, G. Tripepi, F. W. Dekker, and M. van Diepen. "Conducting correlation analysis: Important limitations and pitfalls". In: *Clinical Kidney Journal* 14.11 (May 2021), pages 2332–2337. ISSN: 2048-8505. DOI: 10.1093/ckj/sfab085. eprint: https://academic.oup.com/ckj/article-pdf/14/11/2332/41100015/sfab085.pdf. URL: https://doi.org/10.1093/ckj/sfab085.

[79] H. Joe. "Relative entropy measures of multivariate dependence". In: *Journal of the American Statistical Association* 84.405 (1989), pages 157–164. ISSN: 01621459. URL: http://www.jstor.org/stable/2289859 (visited on 12/06/2022).

[80] P. V. Johnsen, I. Strümke, S. Riemer-Sørensen, A. T. DeWan, and M. Langaas. *Inferring feature importance with uncertainties in high-dimensional data*. 2021. DOI: 10.48550/ARXIV.2109.00855. URL: https://arxiv.org/abs/2109.00855.

[81] T. Karras. *Network architectures used in the StyleGAN paper*. https://tinyurl.com/3a62xnaa. 2019.

[82] T. Karras, T. Aila, S. Laine, and J. Lehtinen. "Progressive growing of gans for improved quality, stability, and variation". In: *CoRR* abs/1710.10196 (2017). arXiv: 1710.10196. URL: http://arxiv.org/abs/1710.10196.

[83] T. Karras, S. Laine, and T. Aila. "A style-based generator architecture for generative adversarial networks". In: *CoRR* abs/1812.04948 (2018). arXiv: 1812.04948. URL: http://arxiv.org/abs/1812.04948.

[84] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. "Analyzing and improving the image quality of StyleGAN". In: *CoRR* abs/1912.04958 (2019).

[85] S. Khodadadeh, S. Ghadar, S. Motiian, W.-A. Lin, L. Bölöni, and R. Kalarot. "Latent to latent: A learned mapper for identity preserving editing of multiple face attributes in StyleGAN-generated images". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2022, pages 3184–3192.

[86] S. Khodadadian, M. Nafea, A. Ghassami, and N. Kiyavash. *Information theoretic measures for fairness-aware feature selection*. 2021. DOI: 10.48550/ARXIV.2106.00772. URL: https://arxiv.org/abs/2106.00772.

[87] G. Kimeldorf and A. R. Sampson. "Monotone dependence". In: *The Annals of Statistics* 6.4 (1978), pages 895–903. ISSN: 00905364. URL: http://www.jstor.org/stable/2958865.

[88] D. E. King. "Dlib-ml: A machine learning toolkit". In: *Journal of Machine Learning Research* 10 (2009), pages 1755–1758.

[89] K. Kira and L. A. Rendell. "A practical approach to feature selection". In: *Proceedings of the Ninth International Workshop on Machine Learning*. ML92. Aberdeen, Scotland, United Kingdom: Morgan Kaufmann Publishers Inc., 1992, pages 249–256.

[90] J. Klein, S. Bhulai, M. Hoogendoorn, and R. Van der Mei. "Plusmine: Dynamic active learning with semi-supervised learning for automatic classification". In: *2021 IEEE/WIC/ACM International Conference on Web Intelligence* (2021).

[91] O. Koyejo, N. Natarajan, P. Ravikumar, and I. S. Dhillon. "Consistent binary classification with generalized performance metrics". In: *Pro-*

ceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. NIPS'14. Montreal, Canada: MIT Press, 2014, pages 2744–2752.

[92] A. Krizhevsky and G. Hinton. *Learning multiple layers of features from tiny images*. Technical report 0. Toronto, Ontario: University of Toronto, 2009.

[93] W. H. Kruskal. "Ordinal measures of association". In: *Journal of the American Statistical Association* 53.284 (1958), pages 814–861. ISSN: 01621459. URL: http://www.jstor.org/stable/2281954 (visited on 11/01/2022).

[94] M. Kubat, R. C. Holte, and S. Matwin. "Machine learning for the detection of oil spills in satellite radar images". In: *Machine Learning* 30.2/3 (1998), pages 195–215. DOI: https://doi.org/10.1023/a:1007452223027.

[95] M. Kuhn. *caret: Classification and regression training*. R package version 6.0-92. 2022. URL: https://CRAN.R-project.org/package=caret.

[96] K. Kumaran, D. Papageorgiou, Y. Chang, M. Li, and M. Takáč. *Active metric learning for supervised classification*. 2018. arXiv: 1803.10647.

[97] M. B. Kursa and W. R. Rudnicki. "Feature selection with the Boruta package". In: *Journal of Statistical Software* 36 (11 Sept. 2010), pages 1–13. ISSN: 1548-7660. DOI: 10.18637/JSS.V036.I11. URL: https://www.jstatsoft.org/index.php/jss/article/view/v036i11.

[98] T. O. Kvålseth. "Note on Cohen's kappa". In: *Psychological Reports* 65.1 (Aug. 1989), pages 223–226. DOI: https://doi.org/10.2466/pr0.1989.65.1.223.

[99] M. Köstinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof. "Large scale metric learning from equivalence constraints". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pages 2288–2295. DOI: 10.1109/CVPR.2012.6247939.

[100] H. O. Lancaster. "Correlation and complete dependence of random variables". In: *The Annals of Mathematical Statistics* 34.4 (1963), pages 1315–1321. ISSN: 00034851. URL: http://www.jstor.org/stable/2238342.

[101] Y. LeCun, C. Cortes, and C. Burges. "MNIST handwritten digit database". In: *ATT Labs [Online]. Available:* http://yann.lecun.com/exdb/mnist 2 (2010).

[102] B. van Leeuwen, A. Gansekoele, J. Pries, E. van de Bijl, and J. Klein. "Explainable kinship: A broader view on the importance of facial features in kinship recognition". In: *International Journal On*

Advances in Life Sciences 14.3 & 4 (2022), pages 89 –99. ISSN: 1942-2660. URL: http://www.iariajournals.org/life_sciences/.

[103] 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". In: *IEEE Access* 6 (2018), pages 12060–12073. DOI: 10.1109/ACCESS.2017.2787719.

[104] X. Li, Y. Wang, S. Basu, K. Kumbier, and B. Yu. "A debiased MDI feature importance measure for random forests". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[105] S. Lipovetsky and M. Conklin. "Analysis of regression in game theory approach". In: *Applied Stochastic Models in Business and Industry* 17.4 (2001), pages 319–330. DOI: https://doi.org/10.1002/asmb.446. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/asmb.446. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/asmb.446.

[106] Z. C. Lipton, C. Elkan, and B. Naryanaswamy. "Optimal thresholding of classifiers to maximize F1 measure". In: *Machine Learning and Knowledge Discovery in Databases*. Edited by T. Calders, F. Esposito, E. Hüllermeier, and R. Meo. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pages 225–239. ISBN: 978-3-662-44851-9.

[107] Y. Y. Lu, Y. Fan, J. Lv, and W. S. Noble. *DeepPINK: Reproducible feature selection in deep neural networks*. 2018. DOI: 10.48550/ARXIV.1809.01185. URL: https://arxiv.org/abs/1809.01185.

[108] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. "From local explanations to global understanding with explainable AI for trees". In: *Nature Machine Intelligence* 2.1 (Jan. 2020), pages 56–67. ISSN: 2522-5839. DOI: 10.1038/s42256-019-0138-9. URL: https://doi.org/10.1038/s42256-019-0138-9.

[109] S. M. Lundberg, G. G. Erion, and S.-I. Lee. *Consistent individualized feature attribution for tree ensembles*. 2018. DOI: 10.48550/ARXIV.1802.03888. URL: https://arxiv.org/abs/1802.03888.

[110] S. M. Lundberg and S.-I. Lee. "A unified approach to interpreting model predictions". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pages 476–4777. ISBN: 9781510860964.

[111] S. M. Lundberg et al. “Explainable machine-learning predictions for the prevention of hypoxaemia during surgery”. en. In: *Nat Biomed Eng* 2.10 (Oct. 2018), pages 749–760.

[112] C. D. Manning, P. Raghavan, and H. SchÃ¼tze. *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008. ISBN: 978-0-521-86571-5. URL: http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

[113] B. Matthews. “Comparison of the predicted and observed secondary structure of T4 phage lysozyme”. In: *Biochimica et Biophysica Acta (BBA) - Protein Structure* 405.2 (Oct. 1975), pages 442–451. DOI: https://doi.org/10.1016/0005-2795(75)90109-9.

[114] L. McInnes, J. Healy, and S. Astels. *Benchmarking Performance and Scaling of Python Clustering Algorithms*. 2016. URL: https://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html.

[115] L. McInnes, J. Healy, and S. Astels. “hdbscan: Hierarchical density based clustering”. In: *The Journal of Open Source Software* 2.11 (Mar. 2017). DOI: 10.21105/joss.00205. URL: https://doi.org/10.21105%2Fjoss.00205.

[116] M. Merler, N. K. Ratha, R. S. Feris, and J. R. Smith. “Diversity in faces”. In: *CoRR* abs/1901.10436 (2019). arXiv: 1901.10436. URL: http://arxiv.org/abs/1901.10436.

[117] L. Merrick and A. Taly. “The explanation game: Explaining machine learning models using Shapley values”. In: *Machine Learning and Knowledge Extraction*. Edited by A. Holzinger, P. Kieseberg, A. M. Tjoa, and E. Weippl. Cham: Springer International Publishing, 2020, pages 17–38. ISBN: 978-3-030-57321-8.

[118] P. E. Meyer. *infotheo: Information-theoretic measures*. R package version 1.2.0.1. 2022. URL: https://CRAN.R-project.org/package=infotheo.

[119] J. H. Min and C. Jeong. “A binary classification method for bankruptcy prediction”. In: *Expert Systems with Applications* 36.3 (Apr. 2009), pages 5256–5263. DOI: https://doi.org/10.1016/j.eswa.2008.06.073.

[120] C. Molnar, G. König, B. Bischl, and G. Casalicchio. *Model-agnostic feature importance and effects with dependent features – A conditional subgroup approach*. 2020. DOI: 10.48550/ARXIV.2006.04628. URL: https://arxiv.org/abs/2006.04628.

[121] T. F. Móri and G. J. Székely. “Four simple axioms of dependence measures”. In: *Metrika* 82.1 (Jan. 2019), pages 1–16. ISSN: 1435-926X.

DOI: 10.1007/s00184-018-0670-3. URL: https://doi.org/10.1007/s00184-018-0670-3.

[122] G. Muhammad and M. Melhem. "Pathological voice detection and binary classification using MPEG-7 audio features". In: *Biomedical Signal Processing and Control* 11 (May 2014), pages 1–9. DOI: https://doi.org/10.1016/j.bspc.2014.02.001.

[123] A. Mungo. *sklearn-relief*. Python package version 1.0.0b2. Dec. 2017. URL: https://libraries.io/pypi/sklearn-relief.

[124] A. B. Owen and C. Prieur. "On Shapley value for measuring importance of dependent inputs". In: *SIAM/ASA Journal on Uncertainty Quantification* 5.1 (2017), pages 986–1002. DOI: 10.1137/16M1097717. eprint: https://doi.org/10.1137/16M1097717. URL: https://doi.org/10.1137/16M1097717.

[125] W. Palmer and R. Allen. "Note on the accuracy of forecasts concerning the rain problem". In: *U.S. Weather Bureau manuscript* (1949).

[126] O. M. Parkhi, A. Vedaldi, and A. Zisserman. "Deep face recognition". In: *Proceedings of the British Machine Vision Conference (BMVC)*. Edited by M. W. J. Xianghua Xie and G. K. L. Tam. BMVA Press, Sept. 2015, pages 41.1–41.12. ISBN: 1-901725-53-7. DOI: 10.5244/C.29.41. URL: https://dx.doi.org/10.5244/C.29.41.

[127] E. Pasolli, H. L. Yang, and M. M. Crawford. "Active-metric learning for classification of remotely sensed hyperspectral images". In: *IEEE Transactions on Geoscience and Remote Sensing* 54.4 (2016), pages 1925–1939. DOI: 10.1109/TGRS.2015.2490482.

[128] F. Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pages 2825–2830.

[129] N. Pilnenskiy. *ITMO-FS*. Python package version 0.3.3. Aug. 2020. URL: https://pypi.org/project/ITMO-FS/.

[130] 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". In: *Sustainability* 12.6 (2020). ISSN: 2071-1050. DOI: 10.3390/su12062427. URL: https://www.mdpi.com/2071-1050/12/6/2427.

[131] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3rd edition. USA: Cambridge University Press, 2007. ISBN: 0521880688.

[132] J. Pries. *The bp dependency package*. https://github.com/joris-pries/BP-Dependency. 2023.

[133] J. Pries. *The bp feature importance package*. https://github.com/joris-pries/BP-Feature-Importance. 2023.

[134] J. Pries. *The DutchDraw package*. https://github.com/joris-pries/DutchDraw. 2023.

[135] J. Pries, G. Berkelmans, S. Bhulai, and R. van der Mei. *The Berkelmans-Pries Feature Importance method: A generic measure of informativeness of features*. 2023. DOI: 10.48550/ARXIV.2301.04740. URL: https://arxiv.org/abs/2301.04740.

[136] J. Pries, S. Bhulai, and R. van der Mei. "Active pairwise distance learning for efficient labeling of large datasets by human experts". In: *Applied Intelligence* (2023). DOI: 10.1007/s10489-023-04516-5.

[137] J. Pries, S. Bhulai, and R. van der Mei. "Evaluating a face generator from a human perspective". In: *Machine Learning with Applications* 10 (2022), page 100412. ISSN: 2666-8270. DOI: https://doi.org/10.1016/j.mlwa.2022.100412. URL: https://www.sciencedirect.com/science/article/pii/S2666827022000871.

[138] J. Pries, E. van de Bijl, J. Klein, S. Bhulai, and R. van der Mei. *The optimal input-independent baseline for binary classification: The Dutch Draw*. DOI: https://doi.org/10.1111/stan.12297. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/stan.12297. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/stan.12297.

[139] *Python implementation of Fisher score computing attribute importance*. https://www.codestudyblog.com/cs2112pyc/1223230432.html. Accessed: 2022-09-26.

[140] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang. "A survey of deep active learning". In: *ACM Comput. Surv.* 54.9 (Oct. 2021). ISSN: 0360-0300. DOI: 10.1145/3472291. URL: https://doi.org/10.1145/3472291.

[141] A. Rényi. "On measures of dependence". In: *Acta Mathematica Academiae Scientiarum Hungarica* 10.3 (Sept. 1959), pages 441–451. ISSN: 1588-2632. DOI: 10.1007/BF02024507. URL: https://doi.org/10.1007/BF02024507.

[142] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti. "Detecting novel associations in large data sets". In: *Science* 334.6062 (Dec. 2011), pages 518–1524.

[143] M. Rezaei and P. Fränti. "Set-matching methods for external cluster validity". In: *IEEE Trans. on Knowledge and Data Engineering* 28.8 (2016), pages 2173–2186.

[144] M. T. Ribeiro, S. Singh, and C. Guestrin. ""Why should I trust you?": Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pages 1135–1144. ISBN: 9781450342322. DOI: 10.1145/2939672.2939778. URL: https://doi.org/10.1145/2939672.2939778.

[145] R. Rothe, R. Timofte, and L. Van Gool. "Deep expectation of real and apparent age from a single image without facial landmarks". In: *International Journal of Computer Vision* 126.2 (Apr. 2018), pages 144–157. ISSN: 1573-1405. DOI: 10.1007/s11263-016-0940-3. URL: https://doi.org/10.1007/s11263-016-0940-3.

[146] O. Russakovsky et al. "ImageNet large scale visual recognition challenge". In: *International Journal of Computer Vision* 115.3 (Apr. 2015). ISSN: 1573-1405. DOI: 10.1007/s11263-015-0816-y. URL: http://dx.doi.org/10.1007/s11263-015-0816-y.

[147] C. R. de Sá. "Variance-based feature importance in neural networks". In: *Discovery Science*. Edited by P. Kralj Novak, T. Šmuc, and S. Džeroski. Cham: Springer International Publishing, 2019, pages 306–315. ISBN: 978-3-030-33778-0.

[148] A. Saabas. *TreeInterpreter*. Jan. 2021. URL: https://pypi.org/project/treeinterpreter/.

[149] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. "300 Faces in-the-wild challenge: The first facial landmark localization challenge". In: *2013 IEEE International Conference on Computer Vision Workshops*. 2013, pages 397–403.

[150] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen. "Improved techniques for training GANs". In: *Advances in Neural Information Processing Systems*. Edited by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Volume 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf.

[151] J. T. Schaefer. "The critical success index as an indicator of warning skill". In: *Weather and Forecasting* 5.4 (Dec. 1990), pages 570–575. DOI: https://doi.org/10.1175/1520-0434(1990)005<0570:tcsiaa>2.0.co;2.

[152] F. Schroff, D. Kalenichenko, and J. Philbin. "FaceNet: A unified embedding for face recognition and clustering". In: *CoRR* abs/1503.03832 (2015). arXiv: 1503.03832. URL: http://arxiv.org/abs/1503.03832.

[153] S. I. Serengil and A. Ozpinar. "LightFace: A hybrid deep face recognition framework". In: *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE. 2020, pages 23–27. DOI: 10.1109/ASYU50717.2020.9259802.

[154] G. Sergioli, R. Giuntini, and H. Freytes. "A new quantum approach to binary classification". In: *PLoS ONE* 14.5 (May 2019). Edited by A. Lund, e0216224. DOI: https://doi.org/10.1371/journal.pone.0216224.

[155] B. Settles. *Active learning literature survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison, 2009.

[156] B. Settles. "From theories to queries: Active learning in practice". In: *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*. Edited by I. Guyon, G. Cawley, G. Dror, V. Lemaire, and A. Statnikov. Volume 16. Proceedings of Machine Learning Research. Sardinia, Italy: PMLR, May 2011, pages 1–18. URL: https://proceedings.mlr.press/v16/settles11a.html.

[157] H. R. Shahraki, S. Pourahmad, and N. Zare. "K important neighbors: A novel approach to binary classification in high dimensional data". In: *BioMed Research International* 2017 (2017), pages 1–9. DOI: https://doi.org/10.1155/2017/7560807.

[158] L. S. Shapley and A. E. Roth, editors. *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Cambridge [Cambridgeshire] ; New York: Cambridge University Press, 1988. ISBN: 9780521361774.

[159] Y. Shen, C. Yang, X. Tang, and B. Zhou. *InterFaceGAN: Interpreting the disentangled face representation learned by GANs*. 2020. arXiv: 2005.09635.

[160] K. Shin, T. Kuboyama, T. Hashimoto, and D. Shepard. "sCWC/sLCC: Highly scalable feature selection algorithms". In: *Information* 8.4 (2017), page 159.

[161] K. Shmelkov, C. Schmid, and K. Alahari. "How good is my GAN?" In: *Computer Vision – ECCV 2018*. Edited by V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss. Cham: Springer International Publishing, 2018, pages 218–234. ISBN: 978-3-030-01216-8.

[162] K. Simonyan and A. Zisserman. *Very deep convolutional networks for large-scale image recognition*. 2014. arXiv: 1409.1556.

[163] M. Sokolova and G. Lapalme. "A systematic analysis of performance measures for classification tasks". In: *Information Processing*

& Management 45.4 (2009), pages 427–437. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2009.03.002. URL: https://www.sciencedirect.com/science/article/pii/S0306457309000259.

[164] E. Song, B. L. Nelson, and J. Staum. "Shapley effects for global sensitivity analysis: Theory and computation". In: *SIAM/ASA Journal on Uncertainty Quantification* 4.1 (2016), pages 1060–1083. DOI: 10.1137/15M1048070. eprint: https://doi.org/10.1137/15M1048070. URL: https://doi.org/10.1137/15M1048070.

[165] S. Stijven, W. Minnebo, K. Vladislavleva, and N. e. Krasnogor. "Separating the wheat from the chaff: On feature selection and feature importance in regression random forests and symbolic regression". eng. In: (2011). URL: http://lib.ugent.be/catalog/pug01:3198359.

[166] C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn. "Bias in random forest variable importance measures: Illustrations, sources and a solution". en. In: *BMC Bioinformatics* 8 (Jan. 2007), page 25.

[167] E. Štrumbelj and I. Kononenko. "Explaining prediction models and individual predictions with feature contributions". In: *Knowledge and Information Systems* 41.3 (Dec. 2014), pages 647–665. ISSN: 0219-3116. DOI: 10.1007/s10115-013-0679-x. URL: https://doi.org/10.1007/s10115-013-0679-x.

[168] M. Sugiyama and K. M. Borgwardt. "Measuring statistical dependence via the mutual information dimension". In: *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. IJCAI '13. Beijing, China: AAAI Press, 2013, pages 1692–1698. ISBN: 9781577356332.

[169] M. Sundararajan and A. Najmi. "The many Shapley values for model explanation". In: *Proceedings of the 37th International Conference on Machine Learning*. Edited by H. D. III and A. Singh. Volume 119. Proceedings of Machine Learning Research. PMLR, July 2020, pages 9269–9278. URL: https://proceedings.mlr.press/v119/sundararajan20b.html.

[170] G. G. Sundarkumar and V. Ravi. "Malware detection by text and data mining". In: *2013 IEEE International Conference on Computational Intelligence and Computing Research*. IEEE, Dec. 2013, pages 1–6. DOI: https://doi.org/10.1109/iccic.2013.6724229.

[171] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018. ISBN: 0262039249.

[172] G. J. Székely and M. L. Rizzo. "Brownian distance covariance". In: *The Annals of Applied Statistics* 3.4 (Dec. 2009), pages 1236–1265. DOI: 10.1214/09-AOAS312. URL: https://doi.org/10.1214/09-AOAS312.

[173] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. "DeepFace: Closing the gap to human-level performance in face verification". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pages 1701–1708.

[174] A. Tharwat. "Classification assessment methods". In: *Applied Computing and Informatics* 17.1 (Jan. 2021), pages 168–192. DOI: 10.1016/j.aci.2018.08.003. URL: https://doi.org/10.1016/j.aci.2018.08.003.

[175] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. "YFCC100M". In: *Communications of the ACM* 59.2 (Jan. 2016), pages 64–73. ISSN: 1557-7317. DOI: 10.1145/2812802. URL: http://dx.doi.org/10.1145/2812802.

[176] S. Tonekaboni, S. Joshi, K. Campbell, D. K. Duvenaud, and A. Goldenberg. "What went wrong and when? Instance-wise feature importance for time-series black-box models". In: *Advances in Neural Information Processing Systems*. Edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Volume 33. Curran Associates, Inc., 2020, pages 799–809. URL: https://proceedings.neurips.cc/paper/2020/file/08fa43588c2571ade19bc0fa5936e028-Paper.pdf.

[177] C. Veenman, M. Reinders, and E. Backer. "A maximum variance cluster algorithm". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.9 (2002), pages 1273–1280. DOI: 10.1109/TPAMI.2002.1033218.

[178] V. Vezhnevets, V. Sazonov, and A. Andreeva. "A survey on pixel-based skin color detection techniques". In: *IN PROC. GRAPHICON-2003*. 2003, pages 85–92.

[179] H. X. Vinh. *QII tool*. Python package version 0.1.3. Feb. 2019. URL: https://pypi.org/project/qii-tool/.

[180] P. Virtanen et al. "SciPy 1.0: Fundamental algorithms for scientific computing in Python". In: *Nature Methods* 17 (2020). Python package version 1.8.0, pages 261–272. DOI: 10.1038/s41592-019-0686-2.

[181] A. Vlachos. "A stopping criterion for active learning". In: *Computer Speech & Language* 22.3 (2008), pages 295–312. ISSN: 0885-2308. DOI: https://doi.org/10.1016/j.csl.2007.12.001. URL: https://www.sciencedirect.com/science/article/pii/S088523080700068X.

[182] S. Wang and C. Manning. "Baselines and bigrams: Simple, good sentiment and topic classification". In: *Proceedings of the 50th Annual*

Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Jeju Island, Korea: Association for Computational Linguistics, July 2012, pages 90–94. URL: https://www.aclweb.org/anthology/P12-2018.

[183] S. Watanabe. "Information theoretical analysis of multivariate correlation". In: *IBM Journal of Research and Development* 4.1 (1960), pages 66–82. DOI: 10.1147/rd.41.0066.

[184] K. Q. Weinberger and L. K. Saul. "Distance metric learning for large margin nearest neighbor classification". In: *J. Mach. Learn. Res.* 10 (June 2009), pages 207–244. ISSN: 1532-4435.

[185] J. West and C. Bergstrom. *Which face is real?* 2019. URL: http://www.whichfaceisreal.com/learn.html.

[186] C. Wheelus, E. Bou-Harb, and X. Zhu. "Tackling class imbalance in cyber security datasets". In: *2018 IEEE International Conference on Information Reuse and Integration (IRI)*. 2018, pages 229–232. DOI: 10.1109/IRI.2018.00041.

[187] B. D. Williamson and J. Feng. "Efficient nonparametric statistical inference on population feature importance using Shapley values". en. In: *Proc Mach Learn Res* 119 (July 2020), pages 10282–10291.

[188] E. Winter. "The Shapley value". In: *Handbook of Game Theory with Economic Applications*. Edited by R. Aumann and S. Hart. 1st edition. Volume 3. Elsevier, 2002. Chapter 53, pages 2025–2054. URL: https://EconPapers.repec.org/RePEc:eee:gamchp:3-53.

[189] R. Wirth and J. Hipp. "CRISP-DM: Towards a standard process model for data mining". In: *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining* (Jan. 2000). URL: http://cs.unibo.it/~danilo.montesi/CBD/Beatriz/10.1.1.198.5133.pdf.

[190] J. Xu, L. Jin, L. Liang, Z. Feng, D. Xie, and H. Mao. "Facial attractiveness prediction using psychologically inspired convolutional neural network (PI-CNN)". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, pages 1657–1661.

[191] L. Yang, R. Jin, and R. Sukthankar. "Bayesian active distance metric learning". In: *CoRR* abs/1206.5283 (2012). arXiv: 1206.5283. URL: http://arxiv.org/abs/1206.5283.

[192] D. Yoo and I. S. Kweon. "Learning loss for active learning". In: *CoRR* abs/1905.03677 (2019). arXiv: 1905.03677. URL: http://arxiv.org/abs/1905.03677.

[193] W. J. Youden. "Index for rating diagnostic tests". In: *Cancer* 3.1 (1950), pages 32–35. DOI: https://doi.org/10.1002/1097-0142(1950)3:1<32::aid-cncr2820030106>3.0.co;2-3.

[194] G. U. Yule. "On the methods of measuring association between two attributes". In: *Journal of the Royal Statistical Society* 75.6 (May 1912), page 579. DOI: https://doi.org/10.2307/2340126.

[195] Yuxiao Hu, Longbin Chen, Yi Zhou, and Hongjiang Zhang. "Estimating face pose by facial asymmetry and geometry". In: *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.* 2004, pages 651–656.

[196] C. Zahn. "Graph-theoretical methods for detecting and describing gestalt clusters". In: *IEEE Transactions on Computers* C-20.1 (1971), pages 68–86. DOI: 10.1109/T-C.1971.223083.

[197] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang. "Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification". In: *IEEE Transactions on Image Processing* 24.12 (2015), pages 4766–4779. DOI: 10.1109/TIP.2015.2467315.

[198] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: A new data clustering algorithm and its applications". In: *Data Mining and Knowledge Discovery* 1.2 (1997), pages 141–182.

[199] S. Zhou, M. L. Gordon, R. Krishna, A. Narcomey, L. Fei-Fei, and M. S. Bernstein. *HYPE: A benchmark for human eye perceptual evaluation of generative models.* 2019. DOI: 10.48550/ARXIV.1904.01121. URL: https://arxiv.org/abs/1904.01121.

[200] Z. Zhou and G. Hooker. "Unbiased measurement of feature importance in tree-based methods". In: *ACM Transactions on Knowledge Discovery from Data* 15.2 (Jan. 2021). ISSN: 1556-4681. DOI: 10.1145/3429445. URL: https://doi.org/10.1145/3429445.

[201] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. "A comprehensive survey on transfer learning". In: *Proceedings of the IEEE* 109.1 (2021), pages 43–76. DOI: 10.1109/JPROC.2020.3004555.

[202] A. Zien, N. Krämer, S. Sonnenburg, and G. Rätsch. "The feature importance ranking measure". In: *Machine Learning and Knowledge Discovery in Databases.* Edited by W. Buntine, M. Grobelnik, D. Mladenić, and J. Shawe-Taylor. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pages 694–709. ISBN: 978-3-642-04174-7.

Summary

Data. A single word that could contain a lot of information. A driving force behind the internet, decision-making, and scientific discoveries. Since the beginning of my PhD, more than 500,000 research articles that contain this word (in the abstract) have been added to the database of Elsevier (a scientific publisher). Data is *valuable*, as long as you have the right techniques to extract information out of the bits and bytes. *Data science* is the scientific field in which these invaluable tools are developed with the goal to extract information, to understand the data, and to use the acquired knowledge to make predictions. In this dissertation, we strive to solve general problems that are a fundamental yet underexposed part of data science. We question commonly used techniques and develop better alternatives. Our research gives practitioners the means to gain accurate insights and draw meaningful conclusions.

Next, we summarize our research and explain the value of our contributions per chapter. We have tried to limit the use of technical terms to make these parts more accessible. Nevertheless, we also highlight possible future research opportunities for the advanced reader. Science is a never-ending relay race, which is why we hope to inspire the reader to pick up one of our many batons.

Chapter 2: Evaluating a Face Generator from a Human Perspective

Summary:

Chapter 2 is about evaluating *face generators*. These are models, trained with computers, that are able to create images of human faces. They are becoming shockingly good at generating *realistic* pictures, by observing and learning from a limited dataset of facial images. The underlying mechanisms and models are truly fascinating, as they can also be used to generate, for example, new paintings in the style of Van Gogh. It is becoming increasingly hard to distinguish between *authentic* and *computer-generated* images. How 'well' a face generator works, is assessed using *performance measures*. A shortcoming of current measures is that they often measure complex intermediate results, whereas a more *humanlike* macroscopic approach could provide additional insights. In Chapter 2, we have introduced such an approach to answer the following two questions for any face generator: (I) Are general human attributes, such as age and gender, learned and transferred from the data? For example, when the model is trained on a lot of images with brown-haired persons, do the generated images also have brown hair? (II) Are the generated faces actually new or do they belong to existing persons from the data?

Utility:

Our approach showed that a well-known face generator (StyleGAN2) *is* able to learn and transfer the human attributes from the data that is used to train this model (I). Furthermore, new faces *are* generated according to *face recognition* methods (II). It is important for practical applications that the generated faces do not belong to identities from the dataset due to *privacy concerns*. These insights provide a deeper understanding into the performance of face generators. This further strengthens the impressiveness of StyleGAN2. It is important to note that our approach is not limited to a *single* model and could be applied to *any* face generator. More generally, we believe that humans are in the end essential for assessing the performance of *generative* methods. By providing a different outlook, we hope that our research instigates novel approaches that advance the field of generative methods.

Research opportunities:

We believe that it is important, when developing a new face generator, to determine whether the *generated* faces match the *training* images too closely. This can even be evaluated during the training phase by integrating face recognition methods to penalize resemblance. Furthermore, face recognition methods should also be trained using *generated* faces to improve their performance. A generated *identity* could be created by using closely distanced latent points. To produce an image with *specific* human attributes, we observed that *truncation* could be used. However, when a face needs to be generated that looks like someone, we preferably need a method to measure *likeness*. Face recognition methods could be utilized, but these are not trained for this purpose. Finding a good likeness measure would improve practical applicability of face generators.

Chapter 3: Active Pairwise Distance Learning for Efficient Labeling of Large Datasets by Human Experts

Summary:

Generating a *random* face is impressive, but generating a face that *looks like* someone is more difficult. Somehow, the *likeness* should be assessed. In Chapter 3, we lay the foundation for training a likeness model. *Face recognition* methods are traditionally trained using *identity datasets*. This means that for *each* image in the dataset, it is known to which *identity* it belongs. However, *no information* is given about the likeness between two images. A picture of *your identical twin* is dealt with the same as an image of a *total stranger*, as they are both not you. This is undesirable for measuring likeness. Ideally, there should be a dataset where the likeness between each pair of images is determined. Attaining such a dataset is, however, very labor-intensive, as there are many possible pairs. To overcome this problem, we introduce and investigate a novel problem withing the field of *Active Learning* named *active pairwise distance learning* (APDL), where the objective is to gather as much information as possible about all pairwise distances, while only labeling a *limited* number of pairs. The goal is thus to determine an *accurate approximation*, while *reducing* the *workload*. Each round, a pair is considered by an expert that determines the label. Herein lies an important choice and opportunity: Which pair is selected? In Chapter 3, we consider many feature-independent *selection strategies* and evaluate their performance.

Utility:

We have introduced the APDL problem, which is *not limited* to the likeness measure. *Any* distance function can be used. We have established *upper* and *lower bound approximations* using general properties of a distance function. We have also shown how these bounds can be *updated* each time the expert labels an additional pair. We have examined many strategies, which showed that: (I) there are *better* strategies than simply *randomly* selecting a pair and (II) selecting with a specific criterion (*max degree*) consistently performed well. In other words, selecting an instance that was already chosen many times provides a lot of information. We chose to investigate strategies that did not use any *feature values* and used *many* datasets in order to find *general* results. As such, the best performing strategies should be used as *baselines* for future research. Already, using these strategies could save many hours and resources, as the labels can be accurately approximated quicker.

Research opportunities:

Our research is a *pioneering* contribution, which is why we have also extensively discussed many opportunities for fruitful future research in Chapter 3. However, we highlight some of them here. We assumed that the expert was *perfect*, which means that *no mistake* was made in the labeling. However, errors are often easily made in practice, which means that an *imperfect* expert should be considered. We have already provided suggestions how to deal with such an expert, but we believe that assuming a flawed expert is the key to many future use cases. Furthermore, new *complex strategies* and improved *prediction methods* could advance the effectiveness even further. Finally, techniques from APDL could be used to *quantify* the *complexity of a dataset*. If more rounds are necessary to approximate the true distances between each pair, it could indicate that the dataset is more complex.

Chapter 4: The Dutch Draw: Constructing a Universal Baseline for Binary Prediction Models

Summary:

In Chapter 4, we introduce a new *baseline* for *binary* prediction methods. *Binary classification* is a fundamental problem within *data science*, where the goal is to *accurately* predict if an instance belongs either to class *zero* or *one*. As simple as this may seems, it has *many* applications, and it is a

stepping stone for more complex classification problems. To explain why a *baseline* is necessary, we look at the following example. Consider a dataset of images, where some are *fake* and others are *real*. When a model is developed to automatically determine this, we measure how well it is able to do so. An accuracy of 0.9 *seems* high, as this means that the model was correct in 90% of the cases. However, the dataset might be *imbalanced* and consists of 95% real and 5% fake images, which means that *always* predicting real might already result in an accuracy of 0.95. In light of this new information, the model's performance of 0.9 does not seem so good anymore. It is important to have a *baseline result* in order to give perspective and meaning to a performance score. In Chapter 4, we construct a novel baseline, which can be determined theoretically for most common performance measures.

Utility:

Our baseline (*Dutch Draw*) can be *theoretically derived* for most commonly used measures. This eliminates the need for *parameter-tuning* and *training* of the baseline. It is *reproducible*, *simple*, and always *applicable*. It should therefore become a standard practice to use and report the Dutch Draw baseline, as it is an *explainable minimal requirement* for any model. However, this does not exempt any practitioner from using a state-of-the-art model, but it does *simplify* and *improve* the development and evaluation process of a new binary classification method. It should be considered a major warning sign when a model fails to beat our baseline.

Research opportunities:

A natural extension to Chapter 4 would be to investigate the *multiclass* setting, where the number of different classes is not limited to two. We believe that *similar* behavior would be observed. It is, however, likely that the baseline is *more easily beatable*. Furthermore, more performance measures could be investigated in order to find *explicit* theoretical formulations. We were not able to find such a formulation for $G_\theta^{(2)}$, which is why this measure should be examined further. If no direct formulation could be found, *approximation algorithms* could be designed to find the *optimal* baseline quickly.

Chapter 5: The Optimal Input-Independent Baseline for Binary Classification: The Dutch Draw

Summary:

In Chapter 4, we have introduced a baseline for binary prediction models. This baseline shines in *simplicity*, as it does not use *feature values*. Nevertheless, there are other baselines that have this same property. In Chapter 5, we compare these baselines in order to find the 'best' baseline. To do this, we first have to establish what 'best' actually entails. For all commonly used performance measures, we prove that the Dutch Draw baseline is the *optimal baseline*, assuming that the dataset is shuffled randomly. This is a natural assumption, as the order is irrelevant for most datasets.

Utility:

Our findings in Chapter 5 provide a strong argument to use the Dutch Draw baseline over *any other* input-independent baseline, as it achieves the best score. Chapter 5 is therefore a stepping stone for *future research*. An example of this, is our own approach we named the *Dutch Oracle*, which greatly improves the interpretability of a performance score. The Dutch Draw baseline gives a reference point for the performance, but it does not supply additional desirable information when beaten. Namely, when two models achieve a score of 0.7 and 0.8, respectively, and the baseline is 0.4, we can conclude that both models have beaten the baseline. However, we would *ideally* also quantify *how much better* the score of 0.8 is compared to 0.7. This is not as simple as taking the difference, due to *nonlinearity* of performance measures. We want to measure how much 'true knowledge' is acquired. The *Dutch Oracle* weighs an *oracle* that *knows* all labels with the 'best guess' *Dutch Draw baseline*. In this way, we can measure how much weight is contributed to the oracle to achieve a specific score. Chapter 5 shows that we should use the Dutch Draw baseline as 'best guess' in this approach.

Research opportunities:

We have already highlighted a way to use the optimal baseline for another method (the Dutch Oracle). Yet, more opportunities to utilize the optimal baseline should be investigated. The Dutch Draw baseline could be used to *standardize* evaluation measures by incorporating the baseline in the performance measure. Furthermore, a natural extension is to examine

multiclass classification and determine an optimal baseline in a similar fashion.

Chapter 6: The Berkelmans-Pries Dependency Function: A Generic Measure of Dependence between Random Variables

Summary:

Measuring *dependency* is an important technique to gain *insight* into a dataset. Are variables related in some shape or form? Does knowing the values of one variable, give information about the values of another variable? These are all highly *relevant questions* in many research areas (think e.g., medicine, psychology, and finance). It is therefore important to *accurately* measure dependency, as it can lead to *significant discoveries*. Finding factors that signify an illness, depression, or a stock market crash could keep you healthy, happy, and possibly rich. However, commonly used measures have many known *flaws*, making it, for example, possible that something that is *fully* dependent is classified as *independent*. This makes *interpreting* the dependency score difficult. In Chapter 6, we start from scratch by determining a list of *desirable* properties. How would a dependency function behave *ideally*? We show that the commonly used methods do *not* satisfy all requirements. Therefore, we introduce a new dependency function that *does* have all these useful properties, which we prove.

Utility:

As previously discussed, *accurately* measuring dependencies is highly relevant in many research fields. This makes the *BP* dependency function truly impactful on a global scale. It is a powerful tool that should be used by *any* data scientist to find and assess relationships between variables. Instead of simply giving yet another new dependency function, we revised the list of desirable properties and found misconceptions in previous literature. For example, we show that a dependency function should be *asymmetrical*, which was often undisputed. Our novel method eliminates common flaws of previous measures and has many desirable properties. Nevertheless, we do not argue that all other dependency measures should be disregarded. Instead, it is important to know the *limitations* when using a measure. What conclusions *can* and *cannot* be drawn? Additionally, we believe that it is *essential* for the dependency function to formulate what properties are desired and why. This line of reasoning could be used in other domains.

Research opportunities:

A remaining challenge for generally applying the *BP* dependency function is how *continuous* data should be handled. *Kernel density estimation* and *data binning* are two types of approaches, but choice of *parameter values* could result in significantly different dependencies. As such, it should be investigated which parameter selection *algorithms* lead to the most *accurate* dependency score. Additionally, more useful properties of the *BP* dependency function could be derived. A dataset is only a representation of the true underlying distribution. It should therefore be investigated how close the *BP* dependency of the dataset is to the dependency of the underlying distribution. Tight upper and lower bound approximations for the *BP* dependency of the underlying distribution would be very valuable for practical applications.

Chapter 7: The Berkelmans-Pries Feature Importance Method: A Generic Measure of Informativeness of Features

Summary:

Datasets are often used to make *predictions*. How this should be done is an important research area within data science. First, the objective was to make *accurate* predictions, which resulted in many recent success stories. Now, the goal becomes to make predictions that can be *explained*. When important decisions are made by a computer, we must be able to explain why the decision was made. One essential approach is to measure *Feature Importance* (FI), which quantifies the *contribution* of each *feature*. It analyzes which components of the data are *relevant* and *irrelevant* for making accurate predictions. If a feature is important, it is deemed to play a crucial role. This could give useful insights into which features actually matter. It can also be used to check for hidden *biases*. Does race or gender play a role in the predictions? Although this information might not explicitly be in the dataset, it could still indirectly affect other variables (think e.g., postal code). When a feature is *unimportant*, it can be removed from the dataset. This can improve both the *speed* and the *performance* of a model. Many FI methods have been suggested, but it can be hard to *interpret* the results due to many limitations. In Chapter 7, we introduce a new FI method based on *game theory* and the *Berkelmans-Pries dependency function* (see Chapter 6). Furthermore, we prove that our approach has a lot of *useful properties*. For example, previous FI methods have not been tested on *exactness*, as it is hard to find datasets where the exact FI is known. We show that our

approach accurately predicts the FI in some cases where the *ground truth FI* can be derived in an exact manner. Experimentally testing if previous methods adhere to the same properties shows that they fail to do so, which demonstrates the potential of our FI method.

Utility:

The BP-FI has *many* desirable properties, which makes it an extremely useful tool for gaining insights into a dataset. As such, it can and should be used in many different research areas. Additionally, we provide a few cases where the desired FI outcome could be determined in an exact manner. This is an important step forward in this field, as no such datasets were previously suggested. We show that many existing methods do *not* attain the desirable outcome, whereas our FI method does. Chapter 7 is the first time that many FI methods are structurally compared and evaluated.

Research opportunities:

A significant step could be made in finding ways to quickly determine Shapley values for datasets with *many* features or at least identifying good *approximation algorithms* for the *Berkelmans-Pries feature importance* (BP-FI). This would allow BP-FI to be feasibly used in additional applications. We have also observed that having *too few* samples could skew the FI values compared with the true underlying distribution. Ideally, mathematical bounds should be derived to determine how many samples are necessary to approximate the underlying distribution. We have proven many properties of BP-FI, yet we are certain that proving new properties could lead to even more insights. Also, more datasets should be identified where the *desired behavior* of an FI method can be derived. This could strengthen our beliefs in BP-FI or lead to new FI methods that perform even better. This is especially important for *local* FI methods, which could also be considered as the *next frontier*. These methods are particularly useful when a *single* prediction needs to be explained. We have suggested ways to adapt the *global* BP-FI to use as a *local* method. Yet, more datasets with desirable outcomes could quantify the quality of the local explanations.

Final remark

In the introduction (see Chapter 1), we have compared this dissertation with a box of chocolates. It contains many different topics in the field of *data science*, each with a unique taste. We hope that the reader found a *tasty* chocolate and tried to experiment with unknown flavors, acquiring a new taste or even coming up with an own new flavor. To those who yearn for more chocolate, know that in our modern society it can be delivered within ten minutes.