KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Operational, Uncertainty-Aware, and Reliable Anomaly Detection

Lorenzo Perini

Supervisor:
Prof. dr. Jesse Davis

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

March 2024

Operational, Uncertainty-Aware, and Reliable Anomaly Detection

Lorenzo PERINI

Examination committee:
Prof. dr. ir. Emmanuel Van Lil, chair
Prof. dr. Jesse Davis, supervisor
Prof. dr. Matthew B. Blaschko
Prof. dr. ir. Hendrik Blockeel
Prof. dr. ir. Elisa Fromont
(Université de Rennes)
Prof. dr. ir. Peter Flach
(University of Bristol)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

March 2024

Uitgegeven in eigen beheer, Lorenzo Perini, Celestijnenlaan 200A box 2402, B-3001 Leuven (Belgium)

Acknowledgments

> Do not take life too seriously. You will never get out of it alive.
>
> *Elbert Hubbard*

Reflecting on my PhD journey, I must recall the unexpected turns and invaluable support that helped me complete this chapter of my academic life.

The initial twist came during the interview process for my PhD, which in a word I would define as "confusing". Like most students, by the end of my master's thesis, I started applying for several PhD positions around Europe. Although in some (let's say few) cases I got immediately rejected, one day I received an email from Jesse saying "Hi Lorenzo, thanks for your application to our PhD position. We are quite interested in your profile and would like to schedule a time to talk to you." Thinking of having just a nice chat with two professors, I was completely relaxed when I joined the online meeting. But guess what? It turned out to be a real interview, with deep questions on probability, statistics, and algorithms! Obviously, my first reaction was "wait, I did not expect this type of question, should I really answer?" Indeed after the interview, I was quite hopeless about getting the position. Instead, less than five hours later I got an email saying they were impressed by my interview. Great news! Wait, is this for real? How can I get the position with just a one-shot interview? My doubts were increasing every hour and I spent at least a day asking friends how they would interpret a non-ambiguous email. Eventually, I had to ask directly Jesse if my getting a PhD position actually meant me going to the next step of the interview. Obviously, they were offering me the position for real. It was an incredible feeling, despite being like a jump in the dark because I knew nothing about Belgium, let alone the research team and topic. What matters is that, after all, it was worth it. This is why, after spending almost $4 \times 12 + 6$ months at KU Leuven (yes, this is the math side of me), I am deeply thankful to Jesse (my supervisor) and Hendrik, who mostly convinced me to take this path through some follow-up emails.

A few months after my start, I set up a meeting with Matthew to ask him to be involved in my PhD jury. I remember the chat with him was inspiring: he immediately set my goal to publish papers at top-tier conferences, especially NeurIPS. And guess what? I kept pushing myself to improve the quality of my research until, eventually, the last one was accepted at NeurIPS 2023. Moreover, I am thankful to Peter for the incredible philosophical chats we had after some talks and during some walks, which played an important role in shaping my personality. Unfortunately, I did not have the chance to meet Elisa but I am happy she decided to join my PhD jury and I deeply appreciate the kind, positive, and encouraging words she used to describe my PhD achievements. Finally, I want to thank Emmanuel for chairing my defense with a lot of positivity and humor, and for helping me out with the stressful organization of my public defense.

Another life-changing event was my research visit to the University of Helsinki in Finland. It all started thanks to an email shared by Hendrik about the possibility of applying for the Sofina-Boël fellowship, which is a grant for talented researchers in Belgium for a long stay abroad. After consulting with Jesse, I managed to get in contact with Prof. Arto Klami and Prof. Paul Bürkner, who agreed to collaborate on a research proposal. The application was successful and the six months I spent in Helsinki were enlightening. I have learned a lot both on the human aspect (e.g., the importance of a work-life balance, the sense of being a team) and on the research aspect (the Bayesian framework, Variational Inference). Also, I appreciated working with Arto's team which was composed of great PhDs and postdocs who helped me settle down and have fun during my visit.

Moving back to Leuven, I have had the great chance to meet and collaborate with incredibly smart people at DTAI. First of all, Vincent V. guided me from the very beginning by teaching me how to properly do research (e.g., setting up the right experiments). Thanks to Pietro I settled down quite easily in Leuven and found the right motivation to keep up with my research, especially when we had to cheer up each other during the covid pandemic. Coming from mathematics, it was not trivial for me to configure the new laptop and learn how to create a good environment to set up my research. Luckily, Pieter and Arne took the lead and, after an hour-ish of non-sense CS terms, they created an easy-to-explain way to obtain always the environment when connecting via ssh to the local machines. Nonetheless, the new young PhDs Lorenzo, Timo, Can, and Luca have brought a sense of freshness and healthy sports competition into our group: going from claims like "I can beat you on any sport of your choice", to supporting the overrated AC Milan that brings tears more often than joy, to complaining for the few stairs we need to climb to grab a coffee. It needed a lot of hard work but, eventually, I was proud of the Italian group we created: it makes me smile thinking that some days there were more Italians than Belgians during the coffee breaks. At its top, the group included the loud Marco, the chatty Eleonora, the wise Andrea and Gabriele - who always managed to level up the conversation by smoothing out the ignorance of me and Luca - the postdocs Emanuele and Paolo, and Prof. Beppe, in addition to the

others mentioned above. Finally, chatting with the other DTAI members has always been insightful. Our chats during the coffee break at 10am and 4pm covered several controversial topics like (i) what vegetarians/vegans can eat - led by Jaron and Loren -, (ii) real and fun stereotypes about Italians - thanks to Adem and Robin -, (iii) the terrible Belgian weather, which, by the way, is always the same for 10 months every year, and (iv) general complaints on Belgian inefficiencies - thanks to Timo, mainly -, despite in my opinion being well-organized.

More importantly, none of this would have been possible without the support of my girlfriend Lavinia. Living abroad for all these years has not affected our relationship in any way. On the contrary, we have learned to solve each other's problems and that we can count on each other when faced with important decisions to make. Going from Italy back to Belgium after the holidays has been a bit sad, but planning our next travel, activity, and new adventure has always cheered me up. Similarly, handling my family's constant and default question "When are you coming back? Why not earlier?" was tough and challenging, but the celebrations after months spent in Belgium were extremely worth it. I was and will always be looking forward to playing with my little nephew Edo!

Am I forgetting anybody? Right, the chaotic beasts from Florence that I call friends or, better, "brothers" as we have known each other for more than 20 years. I am not sure why they have supported me during these years: maybe because they would benefit from my salary, or because they could take advantage of a free sleeping bed and visit Belgium "low-cost". They even came to visit my place by organizing a trip to Amsterdam (sounds nonsense, right?). We have traveled a lot together during holidays and the optimal recipe for a high-quality journey is that Sandro organizes every single step, including reserving flights, renting cars and hotels, and defining what to visit and for how long. What about the others? Mine and Fabio's role is to pretend to know what we do and, smartly, we have always succeeded at convincing the others that we deeply investigated every choice Sandro made. Monto is clearly the food guy (just have a chat with him!) and is the best person to find the cheapest yet inspiring place to eat (although, it did not go well in Bali eh?). Il maestro is our "creative and crazy" component, who always proposes impossible adventures as long as he can take pictures and post them on socials. Bonni must complain about everything, like the unpopular political parties who need to say something to get visibility. I am quite sure Dario has always found out the target of our holiday a just few days before our departure (even when we went to Morocco, Bali, and Lapland). Finally, if you do not have access to a radio but need one, asking any question to Fabian would make him talk for hours (interesting stories though!). I still do not know how a journey with Angelo will be, but looking forward to having him on board!

Lorenzo Perini

28.03.2024, Leuven

Abstract

Anomaly detection methods aim to identify examples that do not follow the expected behavior. For various reasons, anomaly detection is typically tackled by using unsupervised approaches that assign real-valued anomaly scores based on various heuristics. For instance, one can assume that anomalies fall in low-density regions and compute the negative log-likelihood as anomaly score.

Because anomaly scores are often hard to interpret, practitioners need class labels (i.e., anomaly yes/no) for decision-making. That is, one needs to set a proper decision threshold to flag high-score examples as anomalies. However, finding a threshold requires having access to labeled examples for evaluating the quality of the predicted class labels, which is unfeasible in unsupervised anomaly detection. Moreover, existing literature has focused mainly on measuring the quality of the anomaly scores through ranking-based metrics (e.g., AUROC), which largely ignores the problem of how to derive class predictions. Here, we fill this gap by proposing three novel approaches to transform scores into class predictions.

Given a detector's class predictions, a natural question is: how likely does a prediction change when learning a detector on training data that is subject to slight perturbation? Because unsupervised detectors cannot refine the decision boundary by leveraging labeled examples, they tend to have high uncertainty in predictions. That is, slight changes in the training set often would yield a different decision boundary which, in turn, would flip some test examples' class prediction. This uncertainty makes it hard to deploy a detector in real-world applications as it deteriorates the practitioner's trust in its crucial predictions. Because existing literature largely ignores this problem, we fill this gap by proposing an unsupervised approach to quantify a detector's uncertainty in predictions.

While quantifying uncertainty is essential, practitioners also need a reliable way to assess whether they can trust a detector's prediction. That is, one needs to answer the question: is the detector's uncertainty low enough to rely on its prediction? This falls into the field of Learning with Rejection, where the model is allowed to abstain

(i.e., defer the decision, or "reject" it) when its uncertainty is too high, such that practitioners can trust its output whenever it makes a prediction. Traditionally, learning with rejection approaches rely on evaluating the risk (or, equivalently, the cost) of making mispredictions to design the rejection mechanism, which requires labeled examples. Because no unsupervised method for rejection exists, we fill this gap and propose the first unsupervised anomaly detection algorithm with rejection.

Beknopte samenvatting

Methoden voor anomaliedetectie hebben als doel voorbeelden te identificeren die niet voldoen aan het verwachte gedrag. Om verschillende redenen wordt anomaliedetectie typisch aangepakt door gebruik te maken van ongesuperviseerde methodes die reële anomalie-scores toewijzen op basis van verschillende heuristieken. Men kan bijvoorbeeld aannemen dat anomalieën zich bevinden in gebieden met een lage dichtheid en de negatieve log-likelihood berekenen als anomalie-score.

Omdat anomalie-scores vaak moeilijk te interpreteren zijn, hebben eindgebruikers classificatielabels nodig (d.w.z. anomalie ja/nee) voor besluitvorming. Men moet een geschikte beslissingsdrempel instellen om voorbeelden met hoge scores als anomalieën aan te duiden. Het vinden van zo'n drempel vereist echter toegang tot gelabelde voorbeelden om de kwaliteit van de voorspelde classificatielabels te evalueren, wat onhaalbaar is bij ongesuperviseerde anomaliedetectie. Bovendien heeft de bestaande literatuur zich voornamelijk gericht op het meten van de kwaliteit van de anomalie-scores via op rangschikking gebaseerde metrieken (bijv. AUROC), wat grotendeels het probleem negeert van hoe classificatievoorspellingen moeten worden afgeleid. Hier vullen we deze lacune aan door drie nieuwe methodes voor te stellen om scores om te zetten in classificatievoorspellingen.

Een natuurlijke vraag bij de classificatievoorspellingen van een detector is: hoe waarschijnlijk is het dat een voorspelling verandert wanneer een detector wordt getraind op gegevens die onderhevig zijn aan een lichte verstoring? Omdat ongesuperviseerde detectoren de beslissingsgrens niet kunnen verfijnen door gebruik te maken van gelabelde voorbeelden, hebben ze dikwijls een hoge onzekerheid in voorspellingen. Kleine veranderingen in de trainingsset zouden dus vaak een andere beslissingsgrens opleveren, wat op zijn beurt de classificatievoorspelling van sommige testvoorbeelden zou omkeren. Deze onzekerheid maakt het moeilijk om een detector in praktijksituaties in te zetten, omdat het het vertrouwen van de eindgebruiker in de cruciale voorspellingen aantast. Omdat de bestaande literatuur dit probleem grotendeels negeert, vullen we deze lacune aan door een ongesuperviseerde methode te introduceren om de onzekerheid van een detector in voorspellingen te kwantificeren.

Hoewel het kwantificeren van onzekerheid essentieel is, hebben eindgebruikers ook een betrouwbare manier nodig om te beoordelen of ze op een voorspelling van een detector kunnen vertrouwen. Men moet dus de vraag beantwoorden: is de onzekerheid van de detector laag genoeg om op zijn voorspelling te vertrouwen? Dit valt onder het gebied van Leren met Afwijzing, waarbij het model wordt toegestaan om zich te onthouden van een beslissing (dat wil zeggen, het uitstellen of “afwijzen”) wanneer de onzekerheid te hoog is, zodat eindgebruikers de output kunnen vertrouwen wanneer de detector een voorspelling doet. Traditioneel vertrouwen methodes met afwijzing op het evalueren van het risico (of equivalent, de kosten) van verkeerde voorspellingen om het afwijzingsmechanisme te ontwerpen, wat gelabelde voorbeelden vereist. Omdat er geen ongesuperviseerde methode voor afwijzing bestaat, vullen we deze lacune aan en stellen we het eerste ongesuperviseerde anomaliedetectiealgoritme met afwijzing voor.

List of Abbreviations

tn, tp, fn, fp	True negatives, True positives, False negatives, False positives
AUROC	Area Under the Receiver Operating Characteristic
AUPRC	Area Under the Precision-Recall Curve
PU	Positive and Unlabeled (learning)
SCAR, SAR	Selected Completely At Random, Selected At Random
AL	Active Learning
LR	Learning to Reject
Var, Std	Variance, Standard Deviation
G.R.	Global Ranking
IoT	Internet of Things
KL	Kullback-Leibler divergence
i.i.d.	Independent and Identically Distributed
WT	Wind Turbine
MAE	Mean Absolute Error
W,D,L	Wins, Draws, Loses
DPGMM	Bayesian Gaussian Mixture model with a Dirichlet Process prior

List of Symbols

Ω	sample space
$\mathcal{F}$	σ-algebra over Ω
$\mathbb{P}$	probability measure
D	a (labeled or unlabeled) dataset
N, d	number of training examples and features
X	d-dimensional feature random variable
Y	true class label random variable
$\mathcal{H}, h$	hypothesis space, h true relationship between X and Y
p	probability distribution (if combined with upper case symbols) or probability density function (if combined with lower case symbols)
$\mathbb{E}$	expected value
f	anomaly score function
γ, λ	contamination factor, decision threshold
S	anomaly score random variable
$\hat{\bullet}$	estimated quantity or variable $\bullet$
$d(\circ, \bullet)$	distance between $\circ$ and $\bullet$
k	number of neighbors
K, π_i	number of mixture components and mixture proportion of the i-th component
μ, Σ	mean and covariance matrix of a multivariate normal distribution

O, o	label observation random variable and example for PU Learning
ν, $e(x)$	label frequency and propensity score for the example x
$\mathcal{D}$	domain
$\bullet^S$, $\bullet^T$	whether $\bullet$ belong to the source or target domain
c_{fp}, c_{fn}, c_r	cost values for false positives, false negatives and rejections
g	calibration function
$\mathcal{L}$	loss function
®	rejection symbol
$f_{®}$	model with rejection
$\mathcal{M}$, τ	confidence metric and rejection threshold

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> One anomaly a year, not good. Two anomalies in a year, people start talking. Three anomalies in a year, better start looking for a new job.
>
> *Mike Noskov*

The goal of machine learning is to develop algorithms and mathematical models that enable machines to learn from data in order to make predictions or decisions without being explicitly programmed to perform a specific task [119, 260]. In other words, machine learning aims to build systems that learn how to operate a specific task through data, rather than relying on a person hand-crafting a program to handle each case. Importantly, machine learning models usually improve their performance on the task as they are exposed to larger datasets (i.e., more data), as they can learn the input-output relationship characterizing a broader range of scenarios.

One common machine learning task is to train a model to distinguish between two options (i.e., classes), which is known as binary classification task [137]. The classes are typically named as positive and negative. Traditionally, training such a binary classifier requires first collecting a labeled dataset, which is a set of independent and identically distributed examples representing the expected patterns within each class. By leveraging the labeled examples, a fully supervised binary classifier learns to distinguish between the two classes with the goal of making accurate predictions on *future* examples (i.e., ones where the label is unknown).

However, in some cases, one rather wants to monitor a system's normal condition and detect whenever a collected example deviates from the common behavior. This

field is called *anomaly detection* and aims at identifying instances that differ from the expected data patterns, often referred to as anomalies [36, 107]. Normal examples are usually referred to as the negative class, while anomalies compose the positive class. Consider the simple scenario shown in Table 1.1, where a fictional company monitors the daily production of energy from a wind turbine. For each day, they also monitor the average daily temperature, humidity, wind speed, and solar radiation. A natural question is: does the production of energy over the days follow our expectations? With a deeper inspection, one immediately realizes that, on the sixth day, the produced energy was much lower than it should have with such high wind speed: an anomalous event must have happened to the turbine, arguably due to the low temperature and high humidity [245]. Anomaly detection applications span a broad range of other use cases, such as detecting the alteration of proprietary user data [118], water leaks in stores [188], breakdowns in gas turbines [256], or failures in petroleum extraction [159]. Usually, anomalies are associated with a cost such as a monetary cost (e.g., maintenance, paying for fraudulent purchases) or a societal cost such as environmental damages (e.g., dispersion of petroleum or gas). Hence, detecting anomalies in a timely manner is an important problem.

Table 1.1: An example of an anomalous event over the data collected from a wind turbine. Columns indicate values averaged over a day. The sixth day is anomalous because the energy produced is much lower than expected under seemingly favorable weather conditions.

Day	**Temperature (C°)**	**Humidity (%)**	**Wind Speed ($\mathbf{m\,s^{-1}}$)**	**Solar Radiation ($\mathbf{W\,m^{-2}}$)**	**Energy Produced (kW h)**
1	14	25	0.6	200	55
2	12	55	2.8	180	95
3	11	62	6.1	220	160
4	12	35	4.5	190	145
5	7	30	0.8	170	52
6 (Anomaly)	**2**	**85**	**5.2**	**180**	**57**
7	10	48	4.6	185	143

Although anomaly detection is a binary classification task, unfortunately, flagging anomalies is a challenging task because anomaly detection diverges from the traditional, supervised binary classification setting in four key ways (A1 - A4).

A1. Anomaly detection is mostly an unsupervised or a weakly supervised task. Collecting labels, especially for anomalous events, is a hard task for three reasons. First, the expert may not be able to recognize the anomalies. For instance, if Table 1.1 contained 100+ sensors (i.e., columns), how would you process whether a day has an anomalous production of energy? Second, labeling the anomalies may be a time-consuming process. In fact, due to their rarity, the process of labeling anomalies

necessitates the inspection of a huge number of examples (e.g., 1000+) to encounter an anomaly. Third, anomalies may have never occurred so far in the systems, especially for newly deployed systems (e.g., a new operative wind turbine). In this case, it is impractical and undesirable to voluntarily generate some critical event just to observe anomalous behaviors in the data (i.e., for testing purposes) because this can pose safety and ethical concerns [197].

A2. Anomaly detection is an imbalanced task. Because anomalies are rare events, a dataset often contains a vast majority of normal examples. That is, the dataset is strongly imbalanced (e.g., 99.99% of collected examples are normal). Because of the lack of supervision (A1), a machine learning model may not be able to capture the rare anomalous deviations in the data. That is, the model may learn to always predict the normal class, which defeats the point of anomaly detection. Thus, designing anomaly detection models requires accounting for such class bias. As a result, one often needs to make assumptions about the expected type of anomalous behavior to bias the learning algorithm toward detecting such examples [83].

A3. Anomalous examples are not representative of the anomaly class. The anomalies observed within a dataset may not comprehensively represent all potential anomalous cases that can occur. For instance, a wind turbine's blades can be damaged by sandstorms, thunderbolts, or even some icing events, which would have a different impact on the collected data. Reasonably, one almost never gathers all possible anomaly types, especially if you consider that anomalies may change and evolve over time, based on natural changes in the monitored systems or environments [69].

A4. Labeled anomalies may not help in learning an accurate anomaly detector. Anomalies may be a unique one-off event. For instance, wind turbines positioned in a coastal area can experience an unusually powerful and sudden storm with extreme wind conditions, which, in combination with saltwater exposure and, perhaps, a manufacturing defect, may lead a specific blade to a crack that never repeats again in the future. Learning an anomaly detector using one-off anomalies in the training set may have two negative implications. First, the model may overfit the unique anomalies in the training set and fail to detect more general anomalies that could reasonably occur in different scenarios. Second, the model may develop a bias towards the specific conditions of such anomalies, which potentially leads to mispredictions when faced with variations in the operational conditions [217].

To overcome these challenges, traditional anomaly detection models operate in an unsupervised (or weakly supervised) setting by assigning real-valued anomaly scores to examples based on various heuristics, where higher scores indicate more anomalous examples. Roughly speaking, they exploit specific intuitions on what constitutes an anomaly to quantify how anomalous an example is. For instance, one can intuitively assume that anomalies fall far from the normals and use the distance to the k-th nearest neighbor as anomaly scores. Similarly, one can claim that anomalies fall in low-density

regions and compute the negative log-likelihood through a density estimator as anomaly scores. Overall, these anomaly scores serve as the basis for distinguishing anomalies from normal examples.

The literature on anomaly detection has focused on unsupervised and weakly supervised algorithms, but largely ignored three practical challenges in their application:

Operational anomaly detection. Practitioners cannot make decisions based solely on the anomaly scores because they are not interpretable. For instance, given a test example with anomaly score equal to 2, what decision can you make? Most literature evaluates algorithms using ranking-based performance metrics (e.g., AUROC) that only need anomaly scores and test labels to measure the model performance. Unfortunately, *practitioners lack good means for converting the scores to hard predictions* for correct decision-making. Thus, they often resort to using training labels for such goal [220, 252, 263]. This largely defeats the point of using such unsupervised methods.

The practical question is deciding how to set a decision threshold to convert higher scores to anomaly labels and lower scores to normal labels. One approach is to set the decision threshold such that the proportion of predicted anomalies equals the dataset's *contamination factor*, i.e. the real expected proportion of anomalies. However, existing literature assumes the contamination factor to be known (e.g., from domain knowledge) [92, 253, 70], which is almost never the case in real-world applications. Hence, there is a need for proper contamination factor estimation methods to convert anomaly scores into hard predictions.

Uncertainty-aware anomaly detection. In several real-world applications, practitioners refuse to use anomaly detection models because they do not know how reliable the detector's predictions are. Unsupervised detectors tend to have high uncertainty in predictions because they cannot refine their decision boundary using labeled training examples. That is, slight changes in the training set often result in some (test) prediction being flipped. A natural question arises: *would you make decisions based on a detector's predictions that may flip if trained on a slightly different training set?* Alas, this point is largely ignored by most literature, which mainly targets designing novel anomaly detectors instead of quantifying the uncertainty of the existing ones.

For fully supervised binary classification tasks, the common approach to measuring the model's reliability involves *calibrating* the output scores, i.e. transforming them into accurate probabilities [216, 171, 134, 135, 133]. Unfortunately, this requires having access to labeled data, which is impractical in the anomaly detection context where labels are scarce [129, 163]. Consequently, there is a crucial need for alternative metrics that effectively capture the uncertainty in predictions made by unsupervised anomaly detectors.

Reliable anomaly detection. While quantifying model uncertainty is crucial, its practical application for decision-making remains unclear. *After all, given a model's uncertainty value for a prediction, how can a practitioner decide what action to take?* Because the anomaly detector's predictions are usually crucial, practitioners tend to avoid the risk of making wrong decisions by not trusting the model even when it shows minimal uncertainty. Alas, this defeats the whole point of even designing anomaly detection algorithms.

There is an urgent need to increase the user trust in the detection algorithm. One option is to allow the model to tell when its prediction is unreliable such that the user can manually intervene and make the correct decision. That is, if the model makes a prediction it means that it is likely that such prediction is correct. Alternatively, the model abstains and defers the decision to the user. By paying the cost of limiting the number of predictions and the model's effective usage, practitioners can now trust the model's predictions. The field that investigates this option is called *Learning to Reject* [40, 105, 49]. Traditionally, Learning to Reject methods require having access to labels to (1) measure the risk of the model making mispredictions and (2) decide when such risk is large enough to allow the model to abstain [156, 37]. Unfortunately, unsupervised anomaly detection has no access to labeled data, and no existing literature designs unsupervised methods for learning to reject.

1.1 Dissertation Statement

This dissertation investigates unexplored areas of anomaly detection by addressing the aforementioned gaps in the literature. Specifically, we aim to answer the following three questions:

Q1. How can we estimate the contamination factor using weak or no domain knowledge?

Q2. Can we measure the anomaly detector's uncertainty in predictions without relying on any labeled data?

Q3. Can we leverage such an uncertainty estimation method to improve the reliability of an anomaly detector?

1.2 Contributions

This dissertation addresses its statement through five main contributions.

1.2.1 Contribution 1: Class prior estimation in active positive and unlabeled learning

Sometimes one can introduce the human in the loop and query some specific examples' labels to improve the detection of anomalies. However, experts may not understand the concept of anomaly and, therefore, they may be unable to provide a correct anomaly label when asked to. Here, we assume an active learning setting where the user can only provide the label for normal examples. That is, when the example looks anomalous, the user does not give any label.

This weakly supervised setting falls into the field of Positive and Unlabeled (PU) Learning, which is a weakly supervised setting where only labels belonging to one class (named positive) are given, and the unlabeled examples may belong to either class. PU Learning methods include class prior estimation, i.e. estimating the proportion of normals, which is complementary to the contamination factor. However, most PU Learning methods assume that labels are selected completely at random (SCAR), which does not hold in our setting.

Therefore, we propose CAPe, the first algorithm to estimate the class prior (i.e., 1− contamination factor) by correctly handling the problem of labels acquired via active learning. Additionally, we provide an extensive theoretical analysis of CAPe's convergence to the true class prior when increasing the dataset's size. Practically, we investigate two scenarios: i) the user is a perfect oracle and only provides the normal label to true normals, and ii) the user may make mistakes, i.e. incorrectly labeling some anomalies or not providing the label for true normals. In both scenarios, we perform an extensive experimental analysis.

This contribution is based on the following peer-reviewed conference publication [185]:

> Perini, L., Vercruyssen, V., and Davis, J. Class prior estimation in active positive and unlabeled learning. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence (IJCAI-PRICAI 2020).*

We describe our approach in detail in Chapter 3. The Cython code is available at `https://github.com/Lorenzo-Perini/Active_PU_Learning`.

1.2.2 Contribution 2: Transferring the contamination factor between anomaly detection domains by shape similarity

Real-world anomaly detection often involves monitoring groups of related entities, such as machines, windmill farms, or retail stores. These entities share some common behaviors but also have unique characteristics that impact the data collected. Estimating

the contamination factor individually for each entity by collecting labels, especially when monitoring hundreds of them, is impractical. Existing literature investigates the possibility of transferring anomaly information from one entity (the source domain) to similar entities (the target domain), reducing the need for extensive data labeling. However, none of the existing works transfers the contamination factor.

We propose TrADe, an algorithm for transferring contamination factor from an unlabeled source domain (where it is given) to an unlabeled target domain. The method assumes that if the anomaly score distributions of normal examples are similarly shaped in both domains, the target's contamination factor can be derived from the source's known one. Additionally, we perform an extensive theoretical analysis of TrADe and prove that the estimated target contamination factor converges to its true value under mild assumptions. Finally, we perform an extensive experimental analysis.

This contribution is based on the following peer-reviewed conference publication [188]:

Perini, L., Vercruyssen, V., and Davis, J. Transferring the Contamination Factor between Anomaly Detection Domains by Shape Similarity. In *Proceedings of the Thirty-Six AAAI Conference on Artificial Intelligence (AAAI 2022)*.

We describe our approach in detail in Chapter 4. The Python code is available at `https://github.com/Lorenzo-Perini/TransferContamination`.

1.2.3 Contribution 3: Estimating the contamination factor's distribution in unsupervised anomaly detection

Employing inaccurate estimates of the contamination factor (or, equivalently, of the decision threshold) when converting the anomaly scores into predictions deteriorates the performance of the anomaly detector, which decreases user trust in the detection system. If we could pair such an estimate with an indicator of uncertainty, it could enable us to make more informed decisions. While there are existing anomaly detection methods that employ Bayesian Learning to output distributions, none of them targets the transformation of anomaly scores into predictions.

Here, we propose γGMM, the first algorithm for estimating the contamination factor's (posterior) distribution in unlabeled anomaly detection setups. In this case, we do not require any domain knowledge. We experimentally show that γGMM can output a well-calibrated posterior distribution, and that employing its mean as a deterministic point estimate for the contamination factor results in a better thresholding scheme when compared to adapted baselines.

This contribution is based on the following peer-reviewed conference publication [181]:

Perini, L., Bürkner, P.-C., and Klami, A. Estimating the contamination factor's distribution in unsupervised anomaly detection. In *Proceedings of the Fortieth International Conference on Machine Learning (ICML 2023)*.

We describe our approach in detail in Chapter 5. The Python code is available at `https://github.com/Lorenzo-Perini/GammaGMM`. Moreover, γGMM is fully integrated into PyThresh (link: `https://github.com/KulikDM/pythresh`).

1.2.4 Contribution 4: A theoretical framework for assessing an anomaly detector's example-wise stability

Setting the predictive threshold according to the given contamination factor may result in unreliable predictions because small perturbations of the training set would yield (potentially large) variations in the predictive threshold. That is, slightly changing the training anomaly scores yields a different decision threshold being picked, which in turn flips some test predictions.

We investigate this direction and introduce ExCeeD, the first stability measure of the detector's example-wise uncertainty in predictions. Strictly speaking, ExCeeD quantifies how likely a detector's test prediction would flip if we could slightly change the training set. Additionally, we perform a theoretical analysis of the convergence behavior of ExCeeD, and an experimental analysis to assess whether it captures such uncertainty better than some adapted baselines.

This contribution is based on the following peer-reviewed conference publication [187]:

Perini, L., Vercruyssen, V., and Davis, J. Quantifying the confidence of anomaly detectors in their example-wise predictions. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2020)*.

We describe our approach in detail in Chapter 6. The Python code is available at `https://github.com/Lorenzo-Perini/Confidence_AD`. Moreover, ExCeeD is fully integrated into PyOD [258].

1.2.5 Contribution 5: Unsupervised Anomaly Detection with Rejection

High uncertainty (or equivalently, low confidence) estimates mean that the anomaly detector is likely to make mispredictions. A way to take advantage of such a scenario is to include a reject option in the anomaly detector, which means that the model can abstain from making a prediction when its uncertainty is too high. Traditional methods

for Learning to Reject consist of setting an appropriate threshold on a confidence metric and rejecting the example whenever the model prediction has lower confidence. This has the clear effect of increasing user trust, as the model only predicts whenever the prediction has a high chance of being correct.

Unfortunately, existing learning to reject approaches require fully labeled data, which is not available in most anomaly detection settings. Thus, we fill this gap by proposing RejEx, an approach to perform rejection for anomaly detection in a completely unsupervised manner.

We first develop an unsupervised confidence metric based on ExCeeD's stability values. By theoretically analyzing its properties, we develop an unsupervised method to threshold such a confidence metric. Finally, our method comes with strong guarantees, which we carefully investigate and theoretically prove. A large experimental analysis confirms that using our rejection approach improves the detector's performance.

This contribution is based on the following peer-reviewed conference publication [182]:

Perini, L., and Davis, J. Unsupervised Anomaly Detection with Rejection. In *Proceedings of the Thirty-Seven Conference on Neural Information Processing Systems (NeurIPS 2023)*.

We describe our approach in detail in Chapter 7. The Python code is available at `https://github.com/Lorenzo-Perini/RejEx`.

1.3 Additional publications not included in this dissertation.

This dissertation focuses on a subset of my list of publications. Specifically, it contains only the publications that fit within the context of developing operational, uncertainty-aware, and reliable anomaly detection methods. This section summarizes the additional contributions that do not fit within this context.

Learning from Positive and Unlabeled Multi-Instance Bags in Anomaly Detection

In the multi-instance learning (MIL) setting, instances are grouped into bags, and labels are assigned at the bag level, not for individual instances. A bag is labeled positive if it contains at least one positive instance, while negative bag labels mean all instances in the bag are negative. MIL data is common in contexts like anomaly detection, where labeling is rare and costly. In real-world anomaly detection, only positive labels are

often available, fitting into positive-unlabeled (PU) learning. Despite its utility, there is no dedicated work on PU learning in the multi-instance setting for anomaly detection. We propose a novel method for learning from positive and unlabeled bags in anomaly detection, using an autoencoder as the underlying detector. The method modifies the autoencoder's objective function, introducing a new loss to learn from positive and unlabeled bags. The proposed method is theoretically analyzed and empirically evaluated on 30 datasets, outperforming multiple baselines adapted for this setting.

This contribution has been published at a peer-reviewed conference [186]:

Perini, L., Vercruyssen, V. and Davis, J. Learning from Positive and Unlabeled Multi-Instance Bags in Anomaly Detection. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD 2023)*.

Machine Learning with a Reject Option: A survey

Machine learning models always make a prediction, even when it is likely to be inaccurate. This behavior should be avoided in many decision support applications, where mistakes can have severe consequences. Albeit already studied in 1970, machine learning with rejection recently gained interest. This machine learning subfield enables machine learning models to abstain from making a prediction when likely to make a mistake.

This survey aims to provide an overview on machine learning with rejection. We introduce the conditions leading to two types of rejection, ambiguity and novelty rejection, which we carefully formalize. Moreover, we review and categorize strategies to evaluate a model's predictive and rejective quality. Additionally, we define the existing architectures for models with rejection and describe the standard techniques for learning such models. Finally, we provide examples of relevant application domains and show how machine learning with rejection relates to other machine learning research areas.

This contribution has been published at a peer-reviewed journal [105]:

Hendrickx, K.*, Perini, L.*, Van der Plas, D., Meert, W. and Davis, J. Machine learning with a reject option: A survey. *Machine Learning*, 2024.

Semi-Supervised Isolation Forest for Anomaly Detection

Unsupervised anomaly detectors often rely on exploiting intuitions about what constitutes anomalous behavior to overcome the lack of labeled examples. However, such models are limited by the validity of their intuition because these are not universally true but strongly depend on the application task and given dataset. Thus, one

often tries to improve the detectors' performance by using a semi-supervised approach that exploits a few labeled instances to fix the heuristic assumptions. This work proposes a novel semi-supervised tree ensemble based anomaly detection framework, where an isolation tree is learned by leveraging limited labeled examples together with a majority of unlabeled examples. We compare our proposed approach to several baselines and show that it performs comparably well to the best state-of-the-art neural networks on 6 real-world and 14 benchmark datasets.

This contribution has been published at a peer-reviewed conference [223]:

Stradiotti, L., Perini, L., and Davis, J. Semi-Supervised Isolation Forest for Anomaly Detection. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM 2024)*.

Detecting Evasion Attacks in Deployed Tree Ensembles

Tree ensembles are widely used and powerful models but are vulnerable to evasion attacks, where adversaries craft examples to elicit mispredictions. This undermines model performance and user trust. While existing approaches focus on verifying ensemble robustness during learning, we propose an approach to detect adversarial examples post-deployment. We explore this concept, already investigated for neural networks, in the context of tree ensembles. Our key insight is to analyze the *output configuration* of an unseen example, which corresponds to the set of final leaves reached when processed by each tree in the ensemble. By enumerating the leaves, we posit that adversarial examples obtain unusual output configurations. Following this insight, we assess whether an unseen example is adversarial by measuring the distance between its output configuration and the nearest configuration in a reference set. Experimentally, we show that our approach outperforms existing adversarial detection methods when used on three different tree ensemble learners.

This contribution has been published at a peer-reviewed conference [54]:

Devos, L., Perini, L., Meert, W. and Davis, J. Detecting Evasion Attacks in Deployed Tree Ensembles. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2023)*.

Semi-Supervised Learning from Active Noisy Soft Labels for Anomaly Detection

Anomaly detection often employs a semi-supervised approach with active learning to strategically acquire a small number of labels. However, because anomalies are not always well-understood events, the user may be uncertain about how to label

certain instances. To address this, we suggest allowing users to provide soft labels (probabilistic labels) representing their belief that a queried example is anomalous. These labels are naturally noisy due to people's inability to provide well-calibrated probability instances. To cope with this, we propose using a Gaussian Process to learn from actively acquired soft labels in anomaly detection, leveraging information from nearby examples to mitigate noise. We experimentally evaluate our approach on 21 datasets and show that it outperforms several baselines on the majority of experiments.

This contribution has been published at a peer-reviewed conference [158]:

Martens, T., Perini, L. and Davis, J. Semi-supervised Learning from Active Noisy Soft Labels for Anomaly Detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2023)*.

Multi-domain Active Learning for Semi-supervised Anomaly Detection

Active learning aims to reduce the need for extensive annotated data by smartly acquiring labels during the learning process. While current approaches focus on a single dataset, practical scenarios often involve learning models from multiple datasets, each requiring a separate model. Our study addresses the underexplored multi-domain active learning setting, adopting a multi-armed bandits perspective. The proposed Active Learning Bandits (Alba) method employs bandit strategies to explore and exploit the usefulness of querying labels from different datasets. Evaluation on a retail dataset benchmark in a real-world anomalous resource usage detection case demonstrates Alba's superiority over existing active learning strategies, highlighting the limitations of standard approaches in multi-domain settings.

This contribution has been published at a peer-reviewed conference [236]:

Vercruyssen, V., Perini, L., Meert, W. and Davis, J. Multi-domain Active Learning for Semi-supervised Anomaly Detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2022)*.

The effect of hyperparameter tuning on the comparative evaluation of unsupervised anomaly detection methods

Researchers have proposed various anomaly detection algorithms, but their performance depends heavily on user-defined hyperparameters. However, there is no consensus in the literature on how to set these hyperparameters when comparing algorithms. Common approaches, such as using "default" or optimally tuned settings, are criticized for being either too pessimistic or unrealistically optimistic. To address this, we advocate for tuning hyperparameters on a per-dataset basis using a small validation set.

This approach balances the low cost of labeled data acquisition with fair, sound, and reproducible algorithm comparisons. We establish a theoretical lower bound on the required validation set size and support our proposal with experimental results on 16 datasets.

This contribution has been published at a peer-reviewed workshop [220]:

Soenen, J., Van Wolputte, E., Perini, L., Vercruyssen, V., Meert, W., Davis, J. and Blockeel, H.. The effect of hyperparameter tuning on the comparative evaluation of unsupervised anomaly detection methods. In *Proceedings of the KDD'21 Workshop on Outlier Detection and Description (OOD 2021 Workshop at SIGKDD 2021)*.

A Ranking Stability Measure for Quantifying the Robustness of Anomaly Detection Methods

Naturally occurring variations in the data impact the learning of the anomaly detector, and, thus, which examples it will predict to be anomalous. Ideally, an anomaly detection method should be robust to such small changes in the data. Hence, we introduce a ranking stability measure that quantifies the robustness of any anomaly detector's predictions by looking at how consistently it ranks examples in terms of their anomalousness. Our experiments investigate the performance of this stability measure under different data perturbation schemes. In addition, we show how the stability measure can complement traditional anomaly detection performance measures, such as the area under the ROC curve or average precision, to quantify the behavior of different anomaly detection methods.

This contribution has been published at a peer-reviewed workshop [183]:

Perini, L., Galvin, C., Vercruyssen, V. A Ranking Stability Measure for Quantifying the Robustness of Anomaly Detection Methods. In *Proceedings of the 2nd Workshop on Evaluation and Experimental Design in Data Mining and Machine Learning (EDML 2020 Workshop at ECML-PKDD 2020)*.

How to Allocate your Label Budget? Choosing between Active Learning and Learning to Reject in Anomaly Detection

Anomaly detection aims to identify instances deviating from expected behavior, and is often tackled from an unsupervised perspective. However, the lack of labels makes the anomaly detector have high uncertainty in some regions, which usually results in poor predictive performance or low user trust in the predictions. One can reduce such uncertainty by collecting specific labels using Active Learning (AL), which targets examples close to the detector's decision boundary. Alternatively, one can increase the

user trust by allowing the detector to abstain from making highly uncertain predictions, which is called Learning to Reject (LR). Although both AL and LR need labels, they work with different types of labels: AL seeks strategic labels, which are evidently biased, while LR requires i.i.d. labels to evaluate the detector's performance and set the rejection threshold. Because one usually has a unique label budget, deciding how to optimally allocate it is challenging. Here, we propose a reward-based strategy that, given a budget of labels, decides in multiple rounds whether to use the budget to collect AL labels or LR labels. The strategy measures the expected gain when allocating the budget to either side. We evaluate our approach on 18 benchmark datasets and compare it to some baselines.

This contribution has been accepted (yet not officially published) at the peer-reviewed workshop [184]:

Perini, L., Giannuzzi, D. and Davis, J. How to Allocate your Label Budget? Choosing between Active Learning and Learning to Reject in Anomaly Detection. In *1st AAAI Workshop on Uncertainty Reasoning and Quantification in Decision Making*. arXiv preprint arXiv:2301.02909.

Chapter 2

Background

This chapter provides the necessary background information for the contributions of the dissertation. Specifically, we initially describe the canonical anomaly detection problem (Section 2.1), the existing models (Section 2.2), the common evaluation metrics (Section 2.3), and finally move to the related fields that are central to this dissertation (Section 2.4).

2.1 The Anomaly Detection Problem

Anomaly detection aims at identifying examples that could be indicative of errors, fraud, faults, or other negative, unusual, and unexpected events. These examples, referred to as anomalies, show a behavior that is distinct from the expected or normal patterns. The goal of an anomaly detection system is to automatically detect and flag these anomalies within a dataset, which is valuable in several applications. Section 2.1.1 illustrates some of these applications.

The key aspects of anomaly detection include defining what constitutes an anomalous behavior, which usually depends on the specific dataset and application domain. Anomalies can be seen from statistical, machine learning, analytic, or geospatial perspectives, among others. However, one commonly categorizes anomalies according to their geometrical properties (e.g., how do the feature values of anomalies relate to the normal ones?). This allows researchers to provide a complete taxonomy of anomaly types, which we discuss in Section 2.1.2.

Traditionally, one employs machine learning techniques to detect anomalies. Designing an anomaly detector requires first formalizing the anomaly detection problem, namely the concept of a dataset, the anomaly detection task, and the traditional framework. We formalize these three aspects in Section 2.1.3.

2.1.1 Real-World Applications of Anomaly Detection

Anomaly detection has several real-world applications:

Industry. Anomaly detection is central in the industrial field, due to its capacity to detect machine breakdowns [196] or defects in production [25]. Industrial systems like wind turbines [245], power plants [117], high-temperature energy systems [261], and mechanical equipment [144], are exposed to tremendous stress on a daily basis. Unexpected damage within these systems is associated with high monetary costs as well as possible loss of reputation for the industrial brand [35]. Because such damages are rare events, detecting them can be formulated as an anomaly detection problem. For this task, several papers propose promising data-driven methods for detecting early industrial damage [256, 159].

Finance. Anomaly detection has a significant application in the financial field, especially in the detection and prevention of fraudulent activities [6]. Financial fraud is a persistent challenge, and rapid response is crucial for minimizing its impact. For example, anomaly detection helps identify credit card fraud through irregular transactions, such as unusually high payments or atypical purchases [28, 203]. Additionally, it plays a role in monitoring mobile phone billing for unusual usage patterns [179, 120], automating the identification of potentially fraudulent insurance claims [238, 170], and quickly detecting abnormal market behaviors to prevent market fraud and maintain its integrity [5, 84]. These applications are essential for safeguarding financial security and protecting both consumers and businesses.

Medicine. Anomaly detection has vast applications in the medical field, as it enables the quick identification of irregularities in patient data, which can be the key to early disease detection [230, 96]. For example, by examining medical imaging data such as MRIs, anomaly detection can pinpoint anomalous brain regions, which might be indicative of Alzheimer's disease [191, 18]. In critical care units, it plays a vital role in monitoring patients' vital signs and alerting doctors to deviations from normal parameters in real-time, allowing for immediate interventions [34]. Moreover, anomaly detection assists in managing chronic conditions by continuously tracking patients' health metrics and ensuring timely adjustments to treatment plans [218, 61]. The

integration of anomaly detection into medical practice not only improves patient care but also contributes to more accurate diagnoses, better treatment outcomes, and ultimately, the overall well-being of individuals.

2.1.2 Defining and Characterizing the Anomalies

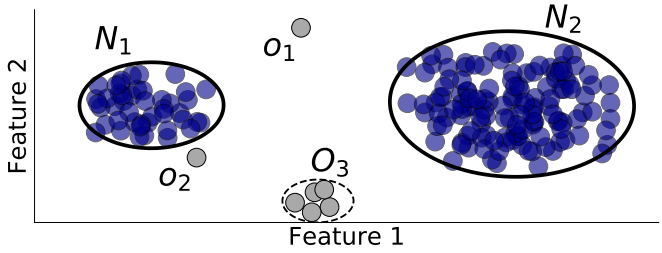

Figure 2.1: Simple 2D toy dataset with 2 clusters of normal examples and 3 types of anomalies: o_1 is a *point* anomaly, o_2 a *contextual* anomaly and O_3 a *collective* anomaly.

The definition of anomalies in academic papers varies depending on the specific field of research and the context of the study. These definitions often revolve around the concepts of *outliers*, *noise*, *novelties*, and more [36]. Although these terms have similar interpretations, they do capture different aspects of the anomaly. For instance, outliers are examples that significantly deviate from the majority of the data. They are observations that are rare or unusual compared to the rest of the data. Outliers are commonly used in fields like finance, where they can represent unusual market behavior, and healthcare, where they might indicate abnormal patient data. Alternatively, in data analysis, noise refers to random variations or errors in data that can distort patterns and relationships. Noisy examples contain irregular and unpredictable values that do not carry meaningful information. Thus, filtering out such noise is crucial for developing an accurate analysis. Finally, novelties are examples that significantly differ from the majority but may not necessarily be erroneous. In some cases, novelties can be of large value, in particular when they show emerging trends, highlight previously undiscovered events or unexplored opportunities. For instance, records of patients with rare pathologies may be of large value for improving a disease detection model.

In this dissertation, we refer to the concept of anomaly as a more general category that includes all previous definitions. Specifically, we define the anomaly as:

An anomaly is an example that does not conform to the expected behavior.

Despite being general, relying on this definition allows us to work on a more abstract level and defer the specific application to the practitioners. From the theoretical perspective, anomalies can also be broadly classified into three types based on their geometric properties: *point anomalies*, *contextual anomalies*, and *collective anomalies* [35]:

- **Point Anomalies.** Point anomalies, also known as individual anomalies, are single examples that deviate significantly from the normal expected behavior. These anomalies do not depend on any specific context and can be detected by analyzing the example in isolation. For instance, in an industrial setting, a point anomaly could be a sudden and unexpected spike in temperature within a chemical reactor during a chemical manufacturing process. This temperature increase may occur due to a temporary sensor malfunction or an external factor like a blocked cooling system [165].
- **Contextual Anomalies.** Contextual anomalies, also known as conditional anomalies, are examples that are considered anomalous within a specific context or condition. These anomalies are identified by taking into account both contextual features (such as time, location, or other relevant conditions) and behavioral features (patterns, behaviors, or attributes) of the data. For instance, if a stock in a stable and well-established company suddenly shows extremely high volatility and rapid price changes without any significant news or events in the broader market, this could be considered a contextual anomaly [165].
- **Collective Anomalies.** Collective anomalies, also known as group anomalies, occur when a collection or group of examples show anomalous characteristics if considered together, while individual examples within the group may appear normal. These anomalies are detected by analyzing the relationships and interactions among examples within the group. For instance, the unexplained increase in hospital admissions in a specific geographic region during a particular time frame can be a collective anomaly [97].

Figure 2.1 shows a visualization of how the three types of anomalies would look on a 2d toy dataset. The point anomaly (o_1) has feature values that are far different from any normals (in each feature). The contextual anomaly (o_2) is close to one cluster of normals and only the combination of both features makes it anomalous. Finally, the collective anomaly (O_3) is a cluster of anomalies that, if taken individually, would not be considered anomalous (due to the other close examples in the group) but, if considered as a unique cluster, their values are far away from any other example.

2.1.3 Defining the Anomaly Detection Problem

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, where Ω is the sample space, $\mathcal{F}$ represents a σ-algebra over Ω and $\mathbb{P}$ is a probability measure. Let $X\colon \Omega \to \mathbb{R}^d$, $Y\colon \Omega \to \{0, 1\}$ be, respectively, a d-dimensional random variable representing a feature vector and a binary random variable representing the true class label. Canonically, $Y = 1$ indicates the anomaly (positive) class, while $Y = 0$ refers to the normal (negative) class.[1]

Definition 1 (Domain)**.** *A domain $\mathcal{D} \coloneqq p(X, Y)$ is defined as a joint probability distribution over the example-label space $\mathbb{R}^d \times \{0, 1\}$.*

In practice, one often only sees a dataset, i.e. an i.i.d. sample drawn from the domain:

Definition 2 (Dataset)**.** *A labeled dataset is a set of N independent and identically distributed (i.i.d) examples $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ drawn from the joint distribution $p(X, Y)$.*

Because most of this dissertation refers to the unlabeled scenario, an unlabeled dataset is just a collection of i.i.d. examples drawn from $p(X)$, as we have no access to their label.

Remark (Unlabeled Dataset)**.** *We refer to an unlabeled dataset with the same symbol $D = \{x_1, \ldots, x_N\}$ and clearly state whether it is labeled (i.e., $(x, y) \in D$) or unlabeled (i.e., $x \in D$).*

The goal of anomaly detection is to learn a detector that maps examples to their class label (anomaly or normal).

Definition 3 (The anomaly detection task)**.** *Given a dataset D, the anomaly detection task consists of finding a function that maps an example x to its label y.*

The unsupervised task is usually treated in two steps: assigning anomaly scores, and transforming the scores into class labels.

Assigning anomaly scores. First, designing an anomaly detection model often refers to finding a measurable map $f\colon \mathbb{R}^d \to \mathbb{R}$ that maps an example x to a real-valued anomaly score, denoted by $s = f(x)$. Such a map f is usually chosen from the hypothesis space $\mathcal{H}$, which is the space of all possible functions satisfying the underlying hypotheses. Essentially, an anomaly score s reflects the degree of anomalousness of the example x: the higher the score the more anomalous the example.

[1] When we deal with Positive and Unlabeled (PU) Learning techniques, we swap the class labels and refer to the normal class as the positive class. This is further motivated in Chapter 3.

Transforming the scores into class labels. Second, one aims at setting a decision threshold $\lambda \in \mathbb{R}$ over the anomaly scores to convert them to hard predictions. In most of the unsupervised (or weakly supervised) problems, setting a correct decision threshold λ is challenging because one cannot set up an optimization problem to tune λ by maximizing the detector's performance. One common approach is to exploit the *contamination factor* $\gamma = \mathbb{P}(Y = 1)$, which measures the expected proportion of anomalies. One can set the decision threshold λ such that $\gamma \times N$ scores are $\geq \lambda$. This guarantees that the number of the detector's predicted anomalies matches the expected number of true anomalies. Such a thresholding approach results optimal in all cases when the detector's anomaly scores for anomalies are higher than for normals.

As a result, a predicted label $\hat{y}$ is assigned to an example x as follows:

$$\hat{y} = \begin{cases} 1 & \text{(anomaly) if } f(x) \geq \lambda; \\ 0 & \text{(normal) if } f(x) < \lambda. \end{cases}$$

Remarks on the notation. In this dissertation, lowercase letters refer to examples while uppercase letters refer to random variables. For instance, we denote by $S = f(X)$ the anomaly score random variable and by $\hat{Y}$ the predicted class label random variable. Moreover, whenever it may be ambiguous, the hat symbol $\hat{\cdot}$ refers to an estimated quantity or variable. Finally, we highlight that $\mathbb{P}$ indicates a probability measure, while p refers either to a random variable's distribution (if used with an uppercase random variable) or to the probability density function (if used with a lowercase example).

2.2 Anomaly Detection Paradigms

The choice of how to design an anomaly detection algorithm depends on two aspects: the level of supervision, and the nature of the anomalies being targeted. That is, one has first to decide between a supervised (i.e., the dataset is fully labeled), a semi-supervised (i.e., the dataset is partially labeled), and an unsupervised (i.e., the dataset has no labels) approach. Then, the choice of the algorithm depends on the underlying intuition of what constitutes an anomalous behavior. In the following text, we briefly discuss some existing anomaly detection models. This discussion is not exhaustive and only focuses on the concepts relevant to this dissertation, which mainly investigates an unsupervised setting.

2.2.1 Supervised Anomaly Detection

Fully-supervised anomaly detection is often impractical due to the high cost and difficulty of collecting large-scale labeled data that includes both normal and anomaly

samples. Arguably, there are no specialized supervised anomaly detection algorithms, and people often use existing classifiers for this purpose such as Random Forest and Neural Networks [4, 233]. However, such models often struggle to detect the unknown ones, as they are not specifically designed for anomaly detection. This is particularly problematic when ground truth labels fail to capture all types of anomalies, limiting the scope of these methods to known anomaly types.

Additionally, supervised approaches face difficulties in achieving satisfactory performance levels due to the challenges associated with addressing class imbalance [45, 86]. Although some loss functions for neural network models have been devised to tackle class imbalance issues (e.g., focal loss [148]), they are often not tailored specifically for anomaly detection tasks.

2.2.2 Semi-Supervised Anomaly Detection

Semi-supervised anomaly detection methods make efficient use of partial labels while maintaining the ability to detect previously unseen anomalies [116, 239]. Instead of relying only on labeled data, these approaches combine unsupervised models with available labels to improve anomaly detection without treating it as a traditional imbalanced classification problem. Roughly speaking, semi-supervised anomaly detectors can be distinguished into four categories based on how they use labels:

1. Incomplete supervision-based semi-supervised anomaly detectors assume all training instances to be normal and learn their distribution to flag out-of-distribution instances as anomalies [39, 263]. For example, LUNAR [85] uses a Graph Neural Network to detect anomalies through a message-passing algorithm. These models can be adapted to use anomaly labels by removing all the labeled anomalies from the training set.

2. Propagation-based semi-supervised anomaly detectors assume that similar/close instances share the same label. Thus, a test instance's prediction depends on how similar/close it is to the given training labeled instances [52]. For example, SSDO [237] and SSkNNo [234] use an unsupervised anomaly detector to initially assign the anomaly score to an instance, and then they update such a score by computing a distance-based correction depending on the neighbors' labels.

3. Loss-based semi-supervised anomaly detectors build an ad-hoc loss function that includes both labeled and unlabeled instances to distinguish between normals and anomalies [259, 73]. For example, SSAD [86] and DeepSAD [206] learn a hypersphere that encapsulates the unlabeled and normally labeled training instances while leaving out the anomalies. Alternatively, Repen [177] uses triplet networks to learn expressive feature representations of the instances, while DevNet [178] leverages a limited number of labeled anomalies to perform end-to-end anomaly score learning.

4. Tree-based semi-supervised anomaly detectors learn the patterns of normal (and potentially anomalous) instances by creating a sequence of splits that isolate the anomalies from the normals. The key assumption is that there exist some features where anomalies can be easily isolated by a split. For instance, HIF [157] uses Isolation Forest [151] to randomly split the instance space, and then it leverages the anomalous labels to measure the proximity to anomalies.

2.2.3 Unsupervised Anomaly Detection

Unsupervised anomaly detection models operate without the need for labeled examples and represent a flexible approach for identifying anomalies across various domains. Because they do not rely on ground truth labels, unsupervised detectors are particularly suitable for scenarios where labeled data is unavailable. To overcome the lack of supervision, unsupervised anomaly detectors exploit data-driven heuristic intuitions on what constitutes an anomaly to assign the anomaly scores.

Normally, unsupervised anomaly detection methods can be categorized according to the nature of the heuristics they use to assign the anomaly scores.[2] Most existing unsupervised methods are implemented in the Python library PyOD [258] and such implementation has been used throughout this dissertation. Here, we provide a simple categorization.

Distance-Based Unsupervised Anomaly Detection

Assumption. *Normal examples occur in dense neighborhoods, while anomalies occur far from their closest neighbors.*

Distance-based anomaly detection relies on measuring the distance or similarity between examples, where the higher the distance the more anomalous the example is. Different measures can be used based on the data type: for continuous attributes, Euclidean distance is popular, while for categorical attributes, matching coefficients or more complex measures can be employed [36].

In the literature, existing methods differ in where they measure the distance from. Here we provide a list of the most common approaches:

[2] Some existing methods, like *Locally Selective Combination of Parallel Outlier Ensembles* (LSCP) [257], employ ensembles of unsupervised models to account for multiple heuristics and capture different types of anomalies.

K-Nearest Neighbor Outlier Detector(kNNO). kNNO [10] assigns the anomaly score for an example x as the distance from its k-th nearest neighbor x':

$$f_{\text{kNNO}}(x) = d(x, x') = d_k(x),$$

where d is usually the Euclidean distance.

Local Outlier Factor (LOF). LOF [29] assigns the anomaly score for an example x as the deviation between its local reachability density (Lrd) and the Lrd of its neighbors, where the Lrd is estimated using the distance to the k-th nearest neighbor d_k:

$$f_{\text{LOF}}(x) = \frac{\sum_{x' \in NN_k(x)} \frac{\text{Lrd}_k(x')}{\text{Lrd}_k(x)}}{|NN_k(x)|}, \quad \text{with} \quad \text{Lrd}_k(x) = \frac{|NN_k(x)|}{\sum_{x' \in NN_k(x)} \max\{d_k(x'), d(x, x')\}}$$

where NN_k is the set of k nearest neighbors of x.

Clustering Based Local Outlier Factor (CBLOF). CBLOF [101] is an extension of LOF that accounts for the cluster structure of the data. It assigns the anomaly score for an example x based on the deviation from the density of its local cluster.

One-Class Support Vector Machines (OCSVM). OCSVM [89] defines a hypersphere (or hyperplane) around the majority of examples and assigns anomaly scores based on the examples' distance to the center of this hypersphere.

Isolation-based anomaly detection using nearest-neighbor ensembles (INNE). INNE [13] uses training subsamples to partition the space into regions and assigns anomaly scores by computing the local distance within the partition.

Statistical Unsupervised Anomaly Detection

Assumption. *Normal examples occur in high-probability regions, while anomalies occur in the low-probability regions.*

Statistical anomaly detection involves learning a probabilistic model, often ad-hoc for normal behavior, and measuring how likely an example is generated by such a model. Examples with a low probability of being generated by the learned model are flagged as anomalies. Specifically, an example's anomaly score relates usually to either its negative log probability density, if the model targets the example density, or the tail probability, if the model targets the cumulative density function. These techniques include both parametric, where the underlying distribution is assumed, and nonparametric methods, which do not rely on such assumption.

In the literature, existing methods differ in how they estimate the density or compute the tail probability. Here we provide a list of the most common approaches:

Kernel Density Estimation for Unsupervised Outlier Detection (KDE). KDE [140] is a parametric method that assumes the examples follow a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ and learns the density $p(x)$ through a traditional kernel density estimator

$$f_{\text{KDE}}(x) \coloneqq \frac{1}{\sqrt{(2\pi)^d |\hat{\Sigma}|}} \exp\left(-\frac{1}{2}(x-\hat{\mu})^\intercal \hat{\Sigma}^{-1}(x-\hat{\mu})\right)$$

where $\hat{\mu}$ and $\hat{\Sigma}$ are the estimated mean and covariance matrix, $|\hat{\Sigma}|$ refers to the determinant of $\hat{\Sigma}$, and $^\intercal$ indicates the transposition.

Outlier detection based on Gaussian Mixture Model (GMM). GMM [205, 4] with K components is an extension of KDE that assumes the examples follow a mixture of K Gaussian distributions and learns the density $p(x)$ as

$$f_{\text{GMM}}(x) \coloneqq \sum_{k=1}^{K} \pi_k \frac{1}{\sqrt{(2\pi)^d |\hat{\Sigma}_k|}} \exp\left(-\frac{1}{2}(x-\hat{\mu}_k)^\intercal \hat{\Sigma}_k^{-1}(x-\hat{\mu}_k)\right)$$

where π_k are the mixing proportions such that $\sum_{k=1}^{K} \pi_k = 1$. For finite mixtures, one typically has a Dirichlet prior over $\{\pi_1, \ldots, \pi_K\}$, but Dirichlet Process (DP) priors allow treating also the number of components as unknown [87].

Histogram-Based Outlier Detection (HBOS). HBOS [82] is a nonparametric method that assumes feature independence to calculate the degree of anomalousness by measuring the example density building histograms.

Lightweight Online Detector of Anomalies (LODA). LODA extends HBOS [189] by using a collection of one-dimensional histograms, where each histogram is constructed on a randomly generated projected space.

Copula Based Outlier Detector (COPOD). COPOD [145] is a nonparametric method that estimates tail probabilities using empirical copula.

Empirical-Cumulative-distribution-based Outlier Detection (ECOD). ECOD [146] estimates tail probabilities by computing the empirical cumulative distribution per dimension and uses the skewness of these empirical distributions to aggregate them.

Spectral Unsupervised Anomaly Detection

Assumption. *Examples can be embedded into a lower dimensional subspace in which normal examples and anomalies appear significantly different.*

Spectral anomaly detection involves mapping the data to a sub-dimensional space to easily identify the anomalies. Such sub-spaces can be either randomly selected

or learned. Each method in this category employs a different heuristic to select the sub-space and measure the anomalousness of the examples. Here we provide a list of the most common approaches:

Isolation Forest (IF) and Deep Isolation Forest (DIF). IF [151] and DIF [247] isolate observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. IF selects the original features while DIF creates an embedded representation of the examples to achieve non-linear splits. Both methods assign anomaly scores using an inverse function of the number of splits needed to isolate an example, where fewer splits indicate more anomalous examples. For instance,

$$f_{\text{IF}}(x) \coloneqq 2^{-\frac{\mathbb{E}[h^*(x)]}{\phi(N)}},$$

where h^* is the height of the tree to isolate x, $\mathbb{E}$ is the expectation over the different trees, and $\phi(N)$ is a normalizing constant value that only depends on the number of training examples N.

Autoencoder (AE) and Variational Autoencoder (VAE). AE [39] and VAE [123] are neural network-based methods that first map examples to a lower dimensional space (AE deterministically, VAE probabilistically), and then reconstruct them as accurately as possible. The goal is to find a lower dimensional representation that still enables accurate reconstruction of the input. Because the network is trained to minimize the reconstruction error on the training set, the higher the reconstruction error for a test example, the more anomalous it is. Thus, they assign the anomaly scores as the reconstruction error. For instance,

$$f_{\text{AE}}(x) = \|x - a_\theta(x)\|^2,$$

where $\|\cdot\|$ is the Euclidean norm, $a_\theta(x)$ is the reconstructed input by the autoencoder with θ as parameters (i.e., the network weights).

Subspace Outlier Detection (SOD). SOD [130] selects the sub-dimensional space for each example by considering the features that significantly deviate from the neighbors in the subspace. It assigns anomaly scores using a distance-based approach to determine how much the object deviates from the neighbors in the subspace.

Principal Component Analysis (PCA) and Kernel Principal Component Analysis (KPCA) Outlier Detector. PCA [215] and KPCA [106] are, respectively, linear and non-linear dimensionality reduction methods using Singular Value Decomposition of the examples (PCA) or a kernel representation of the examples (KPCA) to project them to a lower dimensional space. In this procedure, the covariance matrix of the examples can be decomposed to orthogonal vectors, called eigenvectors, associated with eigenvalues. The eigenvectors with high eigenvalues capture most of the variance in the examples. Anomaly scores are computed using the sum of the projected distance of examples on all eigenvectors.

2.3 Evaluation Metrics for Anomaly Detection

This Section briefly discusses two categories of metrics: *supervised metrics*, which are concerned with evaluating how well an anomaly detector performs in making predictions on labeled examples, and *unsupervised metrics*, which assess the quality of the discovered patterns, clusters, or representations of the data (e.g., stability, robustness).

2.3.1 Supervised Evaluation Metrics

Quantitative evaluation metrics play a crucial role in assessing the performance of anomaly detection models. Among the most commonly used metrics are the confusion matrix, accuracy, F_1 score, and Area Under the Curve (AUC).

Confusion Matrix. The confusion matrix provides a comprehensive view of model performance, breaking down results into true positives tp (anomalies correctly identified), true negatives tn (normals correctly identified), false positives fp (normals misclassified as anomalies), and false negatives fn (anomalies misclassified as normals). Despite giving an excellent high-level overview, the confusion matrix does not allow for an immediate comparison between models. In contrast, single-valued evaluation metrics are more useful for determining how well an anomaly detector performs.

Accuracy. The accuracy is the traditional metric used in binary classification tasks and is computed as the ratio of correct predictions to total predictions

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}.$$

However, the accuracy may not be suitable for imbalanced datasets where anomalies are rare. In fact, a naive anomaly detector that always predicts normal (regardless of the example given as input) would obtain on expectation an accuracy that is as high as 1-γ, where γ is the (low) training contamination factor.

F_1 Score. The F_1 score is particularly well-suited for imbalance classification tasks as it is the harmonic mean of precision and recall:

$$F_1\ score = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{tp}{tp + \frac{1}{2}(fp + fn)}$$

where $Precision = \frac{tp}{tp+fp}$ and $Recall = \frac{tp}{tp+fn}$. The F_1 score is a more appropriate metric for anomaly detection due to its ability to balance between precision (i.e., the accuracy

of positive predictions) and recall (i.e., the true positive rate) [70]. Because the task of anomaly detection is to identify rare and potentially critical events, both the accurate detection of anomalies (high precision) and the capture of most relevant anomalies (high recall) are crucial.

Area Under the Curve Metrics. The Area Under the Curve (AUC) is a significant evaluation metric often used in anomaly detection. Two common AUC measures are the *Area Under the Receiver Operating Characteristic* (AUROC) [95, 94] and the *Area Under the Precision-Recall Curve* (AUPRC) [50, 125], also referred to as Average Precision (AP).

The AUROC measures the area under the Receiver Operating Characteristic (ROC) curve, which is a plot of recall (y-axis) against the false positive rate (x-axis) when varying a decision threshold over the class probabilities or the anomaly scores [161]. Formally, it is computed as

$$AUROC = \int_0^1 Recall\left(FPR^{-1}(x)\right)\, dx$$

where FPR^{-1} is the inverse-image of the false positive rate. Intuitively, the AUROC represents the probability that the model ranks a randomly chosen true anomaly higher than a randomly chosen true normal.

The AUPRC measures the area under the Precision-Recall (PR) curve, which is a plot of recall (x-axis) against precision (y-axis) [68]. The AUPRC quantifies the model's ability to trade off precision for recall as the decision threshold over the class probabilities or the model's scores varies. Formally, it is computed as

$$AUPRC = \int_0^1 Precision\,(Recall)\; dRecall.$$

PR curves and AUPRC are often considered more informative than ROC curves (AUROC) in the presence of class imbalance, as precision and recall provide a more meaningful assessment of a model's performance in situations where the number of anomalies is much smaller than the number of normals [176]. However, Flach and Kull [68] show that PR curves do not satisfy relevant desired properties such as allowing linear interpolation, having a universal baseline, a convex Pareto-front, and an interpretable area. Also, they propose the Precision-Recall-Gain Curve [68] which extends the PR curve to satisfy all previous properties.

Additional remarks on F_1 score vs AUC metrics. Existing anomaly detection literature focuses mostly on the use of AUC (either AUROC or AUPRC) metrics

as the primary evaluation criteria. This preference stems from the fact that AUC metrics only necessitate the model to provide anomaly scores or probabilities for the evaluation, making them particularly advantageous. By using AUC, one effectively bypasses the challenges associated with setting specific decision thresholds on the anomaly scores to predict class labels. The process of thresholding can be challenging, especially in scenarios where the setting is weakly or unsupervised, lacking class labels. However, it's worth noting that this dissertation puts significant emphasis on the task of thresholding anomaly scores. Consequently, when evaluating a detector's performance at specific thresholds, we frequently employ the F_1 score, which helps us assess how effectively the model balances precision and recall, making it a key metric for our research.

2.3.2 Unsupervised Metrics for Anomaly Detection

There exist several unsupervised metrics (i.e., they can be computed without labels) for quantifying detector quality [153]. Roughly speaking, these metrics can be categorized into stand-alone evaluations, which only rely on a single detector's output, and consensus-based, which leverage the agreement between multiple anomaly detectors. In the following, we provide a short description of five existing metrics and refer to the original papers for full details. Note that we employ these metrics in Chapter 7 for hyperparameter tuning (i.e., the rejection threshold) in an unsupervised setting.

Stand-Alone Unsupervised Evaluation Metrics. The initial index proposed for unsupervised outlier detection evaluation is the Internal, Relative Evaluation of Outlier Solutions (**IREOS**) [155], which initially was designed for binary predictions and later was extended to handle anomaly scores [154]. The underlying idea is that anomalies are more easily separated (discriminated) from other training examples than the normals. Thus, a "good" detector effectively identifies highly separable examples, which can be assessed through a nonlinear SVM. Specifically, the IREOS score for a detector f_i is

$$\text{IREOS}(f_i) = \frac{1}{n_\eta} \sum_{l=1}^{n_\eta} \frac{\sum_{j=1}^{N} \phi_{\text{SVM}}(x_j, \eta_l)\omega_j}{\sum_{j=1}^{N} \omega_j}$$

where $\omega_j \in [0, 1]$ is the transformed anomaly score $f_i(x_j)$ into a probability, $\phi_{\text{SVM}}(x_j, \eta_l)$ is the separability of the example x_j estimated by the SVM with kernel bandwidth η_l, and n_η is the number of constant bandwidth values used.
Alternatively, Goix [81] proposed the Mass-Volume (**Mv**) and the Excess-Mass (**Em**) metrics to measure the quality of the anomaly scores. The main idea is that good-quality score distributions have anomalies in the tail. They assume the lower the score the more anomalous, which is the reverse of what we assume but can be easily obtained (e.g., scaling them by taking the inverse or the negative value). Intuitively, the **Mv** measures

the clusteredness of inlier scores, or, equivalently, the compactness of high-density level sets as:

$$\text{Mv}(f_i, \alpha) = \inf_{u \geq 0} |s_{\text{MAX}} - u| \quad \text{s.t.} \quad \frac{1}{N} \sum_{j=1}^{N} \mathbb{1}_{f_i(x_j) \geq u}(x_j) \geq \alpha$$

which computes, for a given $\alpha \in (0, 1)$, the length of the interval between the maximum training score s_{MAX} and the score at the $(1 - \alpha)$-th quantile of the empirical score distribution. Similarly, the **EM** metric identifies as small a u value as possible such that the scores larger than or equal to u are as clustered as possible (to minimize the length of the interval):

$$\text{EM}(f_i, t) = \sup_{u \geq 0} \left\{ \frac{1}{N} \sum_{j=1}^{N} \mathbb{1}_{f_i(x_j) \geq u}(x_j) - t|s_{\text{MAX}} - u| \right\} \quad \text{for } t > 0,$$

where t is set to some value in $[0, \text{EM}^{-1}(f_i, 0.9))$.

Consensus-based Unsupervised Evaluation Metrics. The first consensus-based approach, Unsupervised Disentanglement Ranking (**UDR**), operates under the hypothesis that a robust detector will yield consistent results across different random choices of hyperparameters (HP), unlike a detector that is highly sensitive to the hyperparameter setting [57, 149]. **UDR** involves four steps:

(i) Train M anomaly detectors f_i with E different hyperparameter configurations, for a total of $M \times E$ models;

(ii) For each algorithm f_i, randomly sample (without replacement) $\tilde{E} \leq E$ pairs $(f_i, \text{HP}_{i'})$ sharing the same algorithm f_i and different hyperparameter configurations $\text{HP}_{i'}$;

(iii) Perform $\tilde{E}$ pairwise comparisons between f_i and the sampled models measuring their similarity;

(iv) Aggregate the pairwise comparisons $\text{UDR}_{ii'}$ taking the median $\text{UDR}_i = \text{median}_{i'} \text{UDR}_{ii'}$, and choose the detector f_i with the largest UDR_i.

Essentially, UDR selects the detector with stable results across different HP settings. Finally, the Unsupervised Outlier Model Ensembling (**ENS**) measures the quality of the scores through an iterative scheme designed for unsupervised model selection for ensembles [199, 262]. The key idea is to infer reliable "pseudo ground truth" anomaly scores by aggregating the output of a carefully selected subset of trustworthy detectors. **ENS** iterates over two steps:

(i) Create the pseudo ground truth by aggregating the anomaly scores of the ensemble using weights proportional to the trustworthiness of the models;

(ii) Estimate the trustworthiness of each model based on the ranking similarity between its anomaly scores and the pseudo ground truth.

2.4 Anomaly Detection and Related Fields

Anomaly Detection is a task naturally related to several machine learning fields.

For instance, one often starts with a fully unlabeled dataset and then acquires a few labels by querying an expert. However, experts may not be able to understand the anomalies and often they end up providing labels only for the normal class. This setting falls into the field of Positive and Unlabeled (PU) Learning, where a learner has access only to positive labeled examples and to unlabeled examples (Section 2.4.1).

Additionally, several real-world applications require monitoring multiple assets (e.g., a fleet of wind turbines positioned in different places around the world). Collecting labels for each of the assets is practically infeasible. Therefore, one often resorts to Transfer Learning techniques to transfer the limited domain knowledge given for one dataset to the remaining datasets (Section 2.4.2).

Because anomaly scores are hard to interpret, transforming them into accurate probabilities is crucial in many applications. For instance, if probabilities are accurate, risk-based decisions can be made using techniques from decision-making. Therefore, often one tends to calibrate the model's scores to resemble interpretable probabilities. The set of techniques used for such a goal falls into the field of Calibration (Section 2.4.3).

Given an uncertainty measure (e.g., a calibration method), assessing whether a model is confident enough to make the prediction is an important task. In fact, utilizing the model only when it is likely to make correct predictions has the clear effect of increasing its reliability. This falls into the area of Learning with Rejection, where the model is allowed to abstain from predicting when the risk of misprediction is too high (Section 2.4.4).

2.4.1 Positive and Unlabeled (PU) Learning

The goal of Positive and Unlabeled (PU) Learning is the same as a general binary classification, namely to train a model to distinguish between positive and negative

examples based on their features [21, 56, 62]. However, during the learning phase, only a subset of positive examples is labeled, while none of the negative examples are labeled. That is, the labeled examples must belong to the positive class, while the unlabeled examples can belong to either class.

In PU learning, the canonical example pair (x, y) is extended to be a triple (x, y, o), where $o \in \{0, 1\}$ indicates if the example's label has been observed ($o = 1$) or not ($o = 0$). Therefore, the canonical probability measure $\mathbb{P}$ is naturally extended over possible triples drawn from $\mathbb{R}^d \times \{0, 1\} \times \{0, 1\}$. Moreover, we indicate by $O\colon \Omega \to \{0, 1\}$ the related random variable. With this notation, the key assumption of PU Learning (i.e. the positivity of labels) can be expressed as $\mathbb{P}(Y = 1|O = 1) = 1$.

Labeling Mechanism. The labeled positive examples are selected using an underlying probabilistic mechanism, where each example's label has a probability $e(x) = \mathbb{P}(O = 1|Y = 1, X = x)$ to be observed, which is called *propensity score*. Given the propensity scores, one can link the distribution of the labeled class to the distribution of the positive class as

$$p(x|O = 1) = \frac{e(x)}{\nu} p(x|Y = 1),$$

where $\nu = \mathbb{P}(O = 1|Y = 1) \in [0, 1]$ is the *label frequency*. That is, if one can estimate the propensity score and the label frequency, the true class density can be obtained from the labeled examples using the inverse formula. Moreover, the class prior $\mathbb{P}(Y = 1)$ can be estimated in a straightforward fashion as $\mathbb{P}(O = 1)/\nu$ [21].

Labeling Mechanism Assumptions. Designing PU Learning methods requires making assumptions on the labeling mechanism, which indicates how the positive labels are selected.

1. Selected Completely At Random (SCAR). Traditionally, it is assumed that (positive) labels are selected completely at random (SCAR), meaning that the probability of an example being labeled depends only on its *true* label [19, 42, 38]. This implies that the propensity score is a constant value equal to the label frequency, $e(x) = \nu$. Under the SCAR assumption, the probability of an example being labeled is proportional to its probability of being positive

$$\mathbb{P}(O = 1|x) = \nu \mathbb{P}(Y = 1|x).$$

This enables the use of *non-traditional classifiers*, which are classifiers simply trained to predict O using as positive class $O = 1$ and as negative class $O = 0$.

2. Selected At Random (SAR). More recent work relaxes the SCAR assumption by assuming that the labels are selected at random (SAR). This means that the probability of an example being labeled depends not only on its true label but also on its features [22,

20, 121]. The class prior of a dataset is now computed through the propensity score, where an example's propensity score $e(x)$ can be estimated using insights on the actual labeling mechanism. For instance, in Chapter 3 we estimate the propensity score assuming that the examples are only labeled when they are queried by an active learning strategy [79].

3. Probabilistic Gap. This labeling mechanism assumes that the positive examples that overlap with the negative class distribution are less likely to be labeled [99, 78]. That is,

$$e(x) = \phi(\mathbb{P}(Y = 1|x) - \mathbb{P}(Y = 0|x)) \quad \text{with } \frac{d}{dt}\phi(t) < 0.$$

The main difficulty relies on defining or estimating the function ϕ. One option is to use the observed probabilistic gap $\mathbb{P}(O = 1|x) - \mathbb{P}(O = 0|x)$, which, under mild assumptions, is related to the true gap [99].

2.4.2 Transfer Learning

The main assumption of this dissertation is that anomaly detection tasks benefit from introducing domain knowledge into the setting, such as labels or the exact contamination factor's value. However, if the number of monitored assets is large, how can an expert be able to provide the necessary information for all the assets?

We look into the transfer learning scenario, which aims at transferring information from one domain, called *source domain*, to another domain, called *target domain* [242, 173]. Formally, given a source domain $\mathcal{D}^S$, a target domain $\mathcal{D}^T$, and their datasets D^S and D^T the goal is to improve the target domain model by leveraging the knowledge embedded in the source domain.

In our context, we focus on a specific form of transfer learning, namely domain adaptation, which assumes that the example and label space are identical between the source and target domains [64]. Although the spaces are identical, each domain is characterized by a different probability distribution: $\mathcal{D}^S$ is represented by $p(X^S, Y^S)$, and $\mathcal{D}^T$ by $p(X^T, Y^T)$. A natural way to improve the model performance on the target domain is to align the two distributions, such that one can combine the two (aligned) datasets and learn a model using more data. The joint distributions of the domains can mismatch on three aspects, which define the three common settings (plus combination) for domain adaptation [128, 127].

1. Prior Shift. Prior shift refers to a modification in the class prior distributions between the source and target domains [115, 131]. Decomposing the joint distributions through the chain rule, we have $p(X, Y) = p(X|Y)p(Y)$. The prior shift setting assumes:

$$p(Y^S) \neq p(Y^T) \quad \text{and} \quad p(X^S|Y^S) = p(X^T|Y^T).$$

As a practical example, consider a medical diagnosis system trained on data from a hospital with low disease prevalence (source domain). When deployed to a different hospital with a higher prevalence (target domain), a prior shift occurs in the class prior distribution. This shift necessitates adaptation to ensure the model generalizes effectively in the new environment.

2. Covariate Shift. Covariate shift refers to a modification in the example distribution between the source and target domain [225, 224]. Decomposing the joint distribution using the chain rule in the other way, we have $p(X, Y) = p(Y|X)p(X)$. The covariate shift setting assumes:

$$p(X^S) \neq p(X^T) \quad \text{and} \quad p(Y^S|X^S) = p(Y^T|X^T).$$

As an illustrative example, consider a manufacturing process where a machine learning model is employed for quality control. If the characteristics of raw materials or the production environment change over time, it can lead to a covariate shift, affecting the model's performance. Adapting the model to these shifts ensures reliable and accurate quality assessments.

3. Concept Shift. Concept shift refers to a modification in the class-conditional distributions between the source and target domains [7, 243]. Breaking down the joint distribution as for the covariate shift, the concept shifts setting assumes:

$$p(Y^S|X^S) \neq p(Y^T|X^T) \quad \text{and} \quad p(X^S) = p(X^T).$$

For instance, consider an online fraud detection system trained on transaction data from a global e-commerce platform. When deployed to a new country, where user behavior exhibits different patterns due to cultural practices, a concept shift occurs. If the model is not adapted, it may struggle to identify fraud accurately in the new context, highlighting the importance of addressing concept shift.

Combined shift. Each shift type (prior, covariate, concept) involves a change in joint probability distributions between domains. However, in certain scenarios, multiple shifts may happen together, posing a more challenging domain adaptation problem. Successful transfer learning in such cases requires additional information, such as label data in both domains to fix a concept shift or other assumptions about domain structures.

2.4.3 Calibration

Calibration in binary classification tasks involves aligning predicted probabilities with actual likelihoods, ensuring reliable confidence estimates [216, 90, 23, 171]. A calibrated probability of being anomalous equal to u means that among all instances that get u as a calibrated probability, about $u\%$ of them will be anomalous [132, 231]. Formally, a model f is calibrated if

$$\mathbb{P}(Y = 1 | f(X) = u) = u \quad \forall u \in [0, 1].$$

Well-calibrated probabilities have several benefits, such as that one can combine values consistently or employ standardized decision rules. For example, if the model's probabilities are well-calibrated, one can threshold the probabilities to 0.5 and predict the anomaly class for all examples with higher probability. That is, calibrated probabilities allow adjustment of decision rules to handle varying class priors. More generally, by introducing the costs for mispredictions c_{fp} (false positives) and c_{fn} (false negatives), the optimal decision rule can be easily determined by setting the decision threshold over the anomaly probabilities as $\lambda = \frac{c_{fp}}{c_{fp}+c_{fn}}$ [216].

On the other hand, miscalibrated probabilities often manifest as the model being underconfident or overconfident. In the case of underconfidence, the model believes to be worse at separating classes than it actually is. That is, its predicted probabilities tend to gather towards 0.5. One solution to fix underconfidence involves employing a sigmoidal logistic curve to push predicted probabilities more toward the extremes of the interval $[0, 1]$. On the contrary, overconfidence occurs when the classifier believes it is better at class separation than it actually is. Hence, we need to push the predicted probabilities toward 0.5 to make them less extreme. To address overconfidence, one could design an inverse-sigmoidal map.

Existing calibration methods are traditionally used in a post-hoc fashion, namely they leverage the learned model and a validation set to transform the model's outputs into calibrated probabilities. Formally, a post-hoc calibration map is a function $g\colon \mathbb{R} \to [0, 1]$ such that, $\forall u \in [0, 1]$, it holds that $\mathbb{P}(Y = 1 | g(f(X)) = u) = u$. There is a long literature of approaches [90, 133, 135, 166, 180, 231] for ensuring that this property is obtained and we now briefly describe some prominent approaches: *empirical binning*, *isotonic calibration*, *logistic calibration* and *beta calibration*.

Empirical Binning. Empirical binning is a straightforward method for constructing calibration maps [250, 166]. This approach divides the model's output range into $B \in \mathbb{N}$ non-overlapping bins $\{\mathbb{B}_i\}_{i \leq B}$, where a bin $\mathbb{B}_i \subset [0, 1]$ is a probability range. Then, it links each bin to a (calibrated) probability $a_i \in [0, 1]$:

$$g(s) = a_i \quad \text{for } s \in \mathbb{B}_i.$$

The goal is to find g that maps the non-calibrated scores s falling in the bin $\mathbb{B}_i$ to the empirical frequency. One option is to leverage the calibrated probabilities a_i to minimize the loss

$$\mathcal{L} = \sum_{i}^{B} \left| \mathbb{P}(Y = 1 | \mathbb{B}_i) - a_i \right|.$$

Isotonic Regression. Isotonic regression, a widely employed non-parametric calibration method, seeks the best calibration function among a set of monotonic functions [16, 172, 167]. The method estimates a set of non-decreasing constant segments, corresponding to a set of bins of varying width, with key parameters being bin boundaries and edge values. Similarly to empirical binning, isotonic calibration assigns a constant value to each bin. Unlike empirical binning, isotonic regression dynamically learns bin boundaries from the data, allowing a variable number of bins based on the training set. Formally, one can find the optimum set of bins and calibration function by optimizing the loss

$$\mathcal{L} = \frac{1}{N} \sum_{i}^{N} (g(s_i) - y_i)^2$$

over the monotonic set of edges and values of the bins associated with $g(s_i)$.

Logistic Calibration. Logistic calibration (also known as Platt scaling) proposes transforming the scores into probability estimates using a logistic regression function [219]. The parameters of Logistic calibration include a shape parameter $w \in \mathbb{R}$ and a location parameter $l \in \mathbb{R}$, and has the functional form of

$$g(s) = \frac{1}{1 + \exp(-w \cdot s - l)},$$

where w and l can be estimated by optimizing the training log-likelihood.

Beta Calibration. Beta calibration is a probabilistic generalization of the logistic calibration [134, 135]. Unlike logistic calibration, which assumes a normal distribution of scores within each class, Beta calibration is based on the assumption of two Beta distributions. This introduces a more flexible family of calibration functions with three parameters: two shape parameters $w_1, w_2 \in \mathbb{R}$ and a location parameter $l \in \mathbb{R}$. Thus, g has the functional form of

$$g(s) = \frac{1}{1 + \exp(-w_1 \cdot \log(s) + w_2 \cdot \log(1 - s) - l)}$$

Similarly to the logistic calibration, the parameters w_1, w_2 and l can be estimated by optimizing the log-likelihood on the training set. One can further generalize such an approach by assuming Dirichlet distribution instead of the Beta [133].

Transforming Scores into Class Probabilities in Anomaly Detection. In most anomaly detection applications we lack labeled data with which to train such a calibration model. Thus, three unsupervised methods are widely used to transform scores into class probabilities. The linear and squashing methods map the scores to probabilities using respectively a linear and a sigmoid transformation [74, 237]:

$$g(s) = \frac{s - \min S}{\max S - \min S} \quad \text{(linear)}$$

$$g(s) = 1 - 2^{-\left(\frac{s}{\lambda}\right)^2} \quad \text{(squashing)}$$

where min and max of S are computed using the training set, and λ is the decision threshold set such that $\gamma\%$ of anomaly scores are greater.

The UNIFY method assumes that scores are normally distributed $\mathcal{N}(\mu_S, \sigma_S^2)$ and estimates the class probability through the Gaussian cumulative distribution function [129]

$$g(s) = \max\left\{0, erf\left(\frac{s - \mu_S}{\sqrt{2}\sigma_S}\right)\right\}$$

where erf is the Gaussian error function.

Because of the absence of labels, there is no theoretical guarantee that these methods are calibrated [81].

2.4.4 Learning with Rejection

In the standard supervised setting, one usually assumes that there is an unknown, non-deterministic function $h\colon \mathbb{R}^d \to \{0, 1\}$ that maps the examples to their target label. Given a hypothesis space $\mathcal{H}$ of functions, the goal of a learner is to find a good approximation to h. Typically, this can be done by finding a deterministic model $f \in \mathcal{H}$ with a small expected risk R which is usually approximated using the training data

$$R(f) \coloneqq \int_{\mathbb{R}^d \times \{0,1\}} \mathcal{L}(f(x), y)\, d\mathbb{P}(x, y) \approx \sum_{i=1}^{N} \frac{\mathcal{L}(f(x_i), y_i)}{N}, \tag{2.1}$$

where $\mathcal{L}$ is a suitable loss function such as the squared or zero-one loss.

In learning with rejection, the output space of the model is extended to include a new value ® [105]. This new symbol means that the model abstains from making a prediction. Conceptually, since the predictor f only approximates the true model h, there are likely regions of the space where f systematically differs from h [49, 48]. Specifically, discrepancies between the predictor f and h can be due to inconsistent data (e.g., classes overlapping), insufficient data (e.g., unexplored regions), or even

incorrect model assumptions (e.g., f must be linear while h is not). Therefore, the goal of rejection is to determine such regions to abstain from making likely inaccurate predictions.

At prediction time, a model with rejection $f_{®}\colon \mathbb{R}^d \to \{0, 1, ®\}$ outputs the symbol ® and abstains from making predictions when it is at a heightened risk of making a misprediction, and otherwise returns the predictor's output:

$$f_{®}(x) = \begin{cases} ® & \text{if the prediction is } \textit{rejected}; \\ f(x) & \text{if the prediction is } \textit{accepted}. \end{cases} \tag{2.2}$$

Types of rejection. A predictor is at heightened risk of making mistakes in three different regions of the example space:

R1) when the two class distributions overlap, which refers to the examples x with high class probability $P(Y|X = x)$ for both classes;

R2) close to the predictor's decision boundary, which refers to the examples x with high variance $\textsc{Var}_{D^*}(f(x))$ over possible choices of datasets D^* to train f;

R3) far from the training data, which refers to the examples x with low density $p(x)$.

In the first region, a deterministic predictor is likely to make mistakes because the relationship between the input and the output is non-deterministic. In the second region, the predictor has high instability because its prediction depends on the specific training examples. In the third region, the predictor struggles to make the correct prediction because of the scarcity of training examples.

Based on this intuition, rejection is warranted if an example x falls in one of the three regions (R1-R3). This leads to two types of rejections: *Ambiguity Rejection*, if x falls in a region where the class is ambiguous (R1, R2), and *Novelty Rejection*, if x is a rare example such as a novelty (R3).

Ambiguity rejection allows a model to abstain from predicting an example x in regions where the model f fails to capture the correct relationship h between X and Y [41, 40, 103]. This can happen either due to the probabilistic nature of $Y|X$, which a deterministic predictor f cannot handle, or because the dataset D is insufficient to learn the true function h. The first scenario is often observed as overlapping classes in the feature space. This term is usually irreducible and unrelated to the predictor f. The second scenario occurs when the discrepancy between f and h is large. This is connected to the choice of the hypothesis space $\mathcal{H}$, and the acquired dataset D, which may not allow learning the correct function h.

Novelty rejection allows a model to abstain from predicting examples that are unlikely to occur in the training data [58, 47]. Often, these examples are referred to as novelties. For them, the predictor f does not output a correct prediction because the scarcity of training data similar to x prevents f from learning the correct target y. For instance, the sampling distribution differs from the true distribution and, thus, parts of the feature space are not represented in the data [232, 104, 109]. Similarly, a new distribution may appear after training and these new examples are out-of-distribution [138, 241, 46].

Because the concept of anomaly often overlaps with the concept of novelty (see Section 2.1.2), performing novelty rejection may end up targeting the whole anomaly class for rejection, defeating the goal of using an anomaly detector. Thus, in Chapter 7, *we focus on improving the reliability of a detector performing ambiguity rejection*.

The rejection architecture. At prediction time, the key design decision for a model with rejection is how to structure the relationship between the predictor and rejector. Commonly, researchers have used three architectural principles: (a) a *separated architecture*, where the rejector operates independently from the predictor (e.g., serving as a filter), (b) a *dependent architecture*, where the rejector bases its decision on the output of the predictor (e.g., thresholding a confidence metric), and (c) an *integrated architecture*, where predictor and rejector are integrated into a single model (e.g., rejection is considered as an additional class). Because of its simplicity and effectiveness, in this dissertation, we focus mainly on the dependent architecture.

A canonical example of dependent rejection is when r consists of a pair [confidence $\mathscr{M}_s$, rejection threshold τ] such that an example is rejected if the detector's confidence is lower than the threshold. The model output becomes

$$f_{®}(x) = \hat{y}_{®} = \begin{cases} \hat{y} & \text{if } \mathscr{M}_s > \tau; \\ ® & \text{if } \mathscr{M}_s \leq \tau; \end{cases} \qquad \hat{y}_{®} \in \{0, 1, ®\}.$$

Traditional confidence metrics (such as calibrated class probabilities) quantify how likely a prediction is to be correct. In binary classification tasks, a common approach is to design a confidence metric $\mathscr{M}_s$ as the margin between the two classes' probabilities:

$$\mathscr{M}_s = |\mathbb{P}(Y = 1|s) - \mathbb{P}(Y = 0|s)| = |2\mathbb{P}(Y = 1|s) - 1|.$$

Then, a standard approach to set the rejection threshold is to evaluate different values for τ to find a balance between making too many incorrect predictions because τ is too low (i.e., $\hat{y} \neq y$ but $\mathscr{M}_s > \tau$) and rejecting correct predictions because τ is too high (i.e., $\hat{y} = y$ but $\mathscr{M}_s \leq \tau$) [156, 249].

Both designing a confidence metric using calibrated probabilities $\mathscr{M}_s$ and setting an optimal rejection threshold τ require labels [40] which are unavailable in an unsupervised setting.

Chapter 3

Class prior estimation in active positive and unlabeled learning

Because anomalies are rare and sparse events, understanding whether a collected example is anomalous may be challenging even for a domain expert. For instance, labeling examples collected through several sensors as anomalies might not be a trivial task, especially when the anomalies manifest as unusual configurations of multiple sensors altogether. Therefore, we realistically assume that the expert is not capable of providing a label when queried with real anomalies.[1] Roughly speaking, the expert can provide either a normal label (positive) or no labels.

In this Chapter, we explore the combination of PU learning with *active learning* [2, 229]. That is, the detector initially has access to only unlabeled data, and normal labels are gradually acquired using an active learning strategy [212, 211]. Using active learning introduces the challenge that the SCAR assumption no longer holds as the active learning strategy has a clear bias when selecting examples to be labeled. Thus, estimating the class prior is a hard task because one needs to account for such a biased selection mechanism.

Problem Statement

The problem that this chapter addresses is the following:

[1] In this Chapter, we indicate the normal examples as positives.

Given: An unlabeled dataset D, a non-traditional classifier f trained on PU data, an active learning strategy to obtain l labeled examples;

Do: Estimate the class prior $\alpha = 1 - \gamma$, i.e. the proportion of normal examples.

Contributions of this Chapter.

This chapter analyzes the problem of *estimating the contamination factor from labeled normal and unlabeled data*, where the normal labels are acquired using active learning. We link our task to the traditional PU Learning setting and we refer to the normal class as the positive and to the class prior as the complementary of the contamination factor. That is, the class prior is the proportion of positive examples in the data.

In summary, we make four contributions. First, we show how to estimate the class prior using each example's *propensity score*, which is the probability that an example is selected by the active learning strategy to be labeled, given that it belongs to the positive class. Second, we prove that the estimate of the class prior converges to the true class prior. Third, we propose a method called CAPe (*Class prior estimation in Active Pu lEarning*) for estimating the class prior in practice. Finally, we empirically evaluate CAPe across several anomaly detection datasets. The experiments highlight that CAPe is able to make more accurate estimates of the class prior than current state-of-the-art methods.

The content of this chapter is based on the following publication [185]:[2]

Perini, L., Vercruyssen, V., and Davis, J. Class prior estimation in active positive and unlabeled learning. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence (IJCAI-PRICAI 2020)*, pp. 2915–2921.

3.1 Methodology

Applying the active learning strategy results in a set of l labeled positive examples and $N - l$ unlabeled examples. Subsection 3.1.1 describes how to use this partially labeled data to estimate the *class prior*. However, computing the class prior requires knowing the propensity scores, which is the probability of the example being selected by the active learning strategy to be labeled. Subsection 3.1.2 discusses how to tackle

[2]LP provided the main body of the work (code, theory), VV provided research direction and designing proper experiments, JD assisted in nailing down the right research questions, formalizing and structuring the text.

this issue. Finally, Subsection 3.1.3 shows that if the computed propensity scores are accurate, the estimated class prior theoretically converges to the true class prior.

3.1.1 Estimating the Class Prior

For now, we assume that the propensity scores are known. Intuitively, the class prior can be derived by combining the proportion of examples labeled positive by the user, and the expected proportion of positive examples among the remaining unlabeled examples:

$$\alpha = \mathbb{E}[\mathbb{P}_f(Y = 1|X)] = \mathbb{E}[O|X] + \mathbb{E}\left[(1-O|X)\frac{\mathbb{P}_f(Y = 1|X)\,(1-e(X))}{1-\mathbb{P}_f(Y = 1|X)\,e(X)}\right] \tag{3.1}$$

where $O|X$ is 1 if the example's label has been observed and 0 otherwise, $\mathbb{P}_f(Y = 1|X)$ is the probability that an example belongs to the positive class according to the trained classifier f trained on the PU data, and $e(X)$ is the propensity score defined in Eq. 3.2.

We derived the previous identity from Bekker et al. [22] by applying the expectation on both sides. The main assumption is that f returns *accurate* estimates of the class probabilities. Thus, class prior estimates also depend on the correctness of this assumption. Roughly speaking, a labeled example contributes fully towards the positive class prior, while an unlabeled example only contributes its probability of being positive weighted by its probability of being labeled.

3.1.2 Estimating the Propensity Scores

The propensity score for an example x is the proportion of all datasets containing x that can be drawn from the distribution $p(X)$, in which x's label is observed

$$e(x) = \mathbb{P}(O = 1|x, Y = 1). \tag{3.2}$$

Whether x's label is observed depends on both the dataset and the active learning strategy: whether x is selected to be labeled will depend on whether another more informative unlabeled example (according to the active learning strategy) is present in the dataset. This is further complicated by the fact that, in practice, we only have one dataset from which to estimate propensity scores.

Conceptually an example's propensity score can be computed as:

$$e(x) = \int_{\mathbb{R}^d} \mathbb{P}(O = 1|x, Y = 1, D^*)\, d\mathbb{P}(D^*|x, Y = 1) = \sum_{\{D^* \subset \mathbb{R}^d :\ x \in D^*\}} e(x|D^*) \cdot d\mathbb{P}(D^*|x, Y = 1), \tag{3.3}$$

where the sum is over the infinitely many possible datasets D^* that can be drawn from $\mathbb{R}^d$, $d\mathbb{P}(D^*|x, Y = 1)$ is the infinitesimal probability to draw a specific sample D^*, and $e\,(x|\,D^*)$ is what we call the *grounded propensity score* given the observed dataset D^*. Our key insight is that for a fixed dataset D^*, $e(x|D^*)$ is either 0 or 1: either the example is in the top-l most informative examples of the dataset according to the active learning strategy (1) or it is not (0).[3]

Next, we tackle the problem of summing over infinitely many possible datasets. This can be solved by decomposing the problem into: (1) an inner loop summing over all subsets of $\mathbb{R}^d$ with a given cardinality m, and (2) an outer loop summing over all possible cardinalities.

Inner loop: summing over the subsets. Considering only those subsets of $\mathbb{R}^d$ with cardinality m, we define:

$$\overline{e_m(x)} := \sum_{\{D^* \subset \mathbb{R}^d, |D^*|=m, x \in D^*\}} e(x|D^*) \cdot d\mathbb{P}(D^*|x, Y = 1). \tag{3.4}$$

Taking the sum over all possible subsets of cardinality m poses two questions. First, is this sum actually countable, even if the set is apparently *uncountable*? And second, if so, how can we compute it? The countability of the sum is proven using the following theorem.

Theorem 1. *Let I be any set, $v\colon I \to [0, +\infty)$. Let's define*

$$\sum_{i \in I} v(i) = \sup\left\{ \sum_{i \in J} v(i)\colon J \subset I, |J| < +\infty \right\}.$$

Then, if $\sum_{i \in I} v(i) < +\infty$, the set

$$A = \{i \in I\colon v(i) \neq 0\}$$

is at most countable.

Proof. Let's consider $\varepsilon > 0$ and $A_\varepsilon = \{i \in I\colon v(i) > \varepsilon\}$. Without loss of generality, we suppose that $|A_\varepsilon| = +\infty$ and that its cardinality is countable. Then there exists a sequence $\{x_n\}_{n \in \mathbb{N}} \subseteq A_\varepsilon$ such that $v(x_n) > \varepsilon$ for all $n \in \mathbb{N}$. So, since ε is a constant, the inequality

$$\sum_{n=1}^{\infty} v(x_n) > \sum_{n=1}^{\infty} \varepsilon = +\infty$$

[3] That is under the assumption that there is no randomization in the active learning strategy or learning algorithm.

holds. This leads to a contradiction:

$$\sup\left\{\sum_{i\in J} v(i)\colon J\subset I, |J|<+\infty\right\} = +\infty.$$

As a result the set A_ε is finite. Because of the arbitrary choice of ε, let's choose $\varepsilon = \frac{1}{t}$, for $t \in \mathbb{N}$. Now it is evident that

$$A = \{i \in I\colon v(i) \neq 0\} = \bigcup_{t\in\mathbb{N}} A_{\frac{1}{t}}$$

is countable, since it is a countable union of finite sets. □

To answer the second question, the sum over all possible subsets with cardinality m can be *approximated* through a sequence of u subsets. Since the sum is actually countable, there exists a sequence of sets $D^*_{(m_1)}, \ldots, D^*_{(m_u)}, \ldots$ with non-zero values such that

$$\sum_{i=m_1}^{m_u} e\left(x|D^*_{(i)}\right)\cdot d\mathbb{P}\left(D^*_{(i)}|x, Y=1\right) \xrightarrow{u\to\infty} \overline{e_m(x)}.$$

Subsections 3.2.1 and 3.2.2 explain how to compute this sequence in practice.

Outer loop: summing over the cardinalities. Next, we need to sum over all possible cardinalities to arrive at the propensity score for an example x. So, we define

$$e_m(x) \coloneqq \sum_{j=1}^{m} \overline{e_j(x)}$$

where $\overline{e_j(x)}$ is defined in Eq. 3.4. This sequence converges to the actual propensity score for m going to $+\infty$.

Proof.

$$\begin{aligned}
\sum_{j=1}^{m}\overline{e_j(x)} &= \sum_{j=1}^{m}\sum_{\{D^*\subset\mathbb{R}^d\colon |D^*|=j,\, x\in D^*\}} e(x|D^*)\cdot d\mathbb{P}(D^*|x, Y=1)\\
&= \sum_{\{D^*\subset\mathbb{R}^d\}} e(x|D^*)\cdot d\mathbb{P}(D^*|x, Y=1)\sum_{j=1}^{m}\mathbb{1}_{\{|D^*|=j,\, x\in D^*\}}(D^*)\\
&= \sum_{\{D^*\subset\mathbb{R}^d\}} e(x|D^*)\cdot d\mathbb{P}(D^*|x, Y=1)\mathbb{1}_{\bigcup_{j=1}^{m}\{|D^*|=j,\, x\in D^*\}}(D^*)\\
&\longrightarrow \sum_{\{D^*\subset\mathbb{R}^d,\, x\in D^*\}} e(x|D^*)\cdot d\mathbb{P}(D^*|x, Y=1) \quad \text{for } m\to\infty.
\end{aligned}$$

□

So, theoretically, it is possible to determine $\widetilde{m}$ and some small error ε such that, for $\widetilde{m}$ "large enough",

$$\|e_{\widetilde{m}}(x) - e(x)\| = \left\| \sum_{j=1}^{\widetilde{m}} \overline{e_j(x)} - e(x) \right\| < \varepsilon.$$

Practical computation of the propensity scores. In practice, in order to compute the outer and inner loop of the sum in Eq. 3.1.2, we need to choose the parameters m and u. Their values, however, are restricted by the observed dataset D: m is maximally equal to N and the u subsets can only be drawn from D. Moreover, computing the inner loop over all possible subsets of a certain cardinality is prohibitively expensive. For instance, if D contains 2000 examples and the cardinality is 1000, we would need to loop over $> 10^{600}$ possible subsets. We circumvent this issue by applying standard counting techniques on the available dataset to directly estimate $\overline{e_m(x)}$. Then, through an average approximation of the probabilities, the inner loop can be completely avoided. Section 3.2 explains this in detail.

3.1.3 Convergence to the True Class Prior

If we obtain an accurate estimate of the propensity scores, the convergence of the estimated class prior to the true class prior follows from the following theorem:

Theorem 2. *Assume that there exists a sequence $e_m(x)$ of functions which converges to the propensity score $e(x)$ for all $x \in D$. Then, given the sequence of class priors*

$$\alpha_m := \mathbb{E}\left[O + (1 - O) \frac{\mathbb{P}_f(Y = 1|X)\,(1 - e_m(X))}{1 - \mathbb{P}_f(Y = 1|X)\,e_m(X)} \right],$$

the following result holds

$$\alpha_m \longrightarrow \alpha \qquad \text{for } m \to \infty.$$

Proof. The hypothesis means that

$$\lim_{m \to \infty} e_m(x) = e(x) \qquad \forall x \in D.$$

Then,

$$\begin{aligned}
\lim_{m\to\infty} \alpha_m &= \lim_{m\to\infty} \mathbb{E}\left[O + (1-O)\frac{\mathbb{P}_f(Y=1|X)\,(1-e_m(X))}{1-\mathbb{P}_f(Y=1|X)\,e_m(X)}\right] \\
&= \mathbb{E}\left[\lim_{m\to\infty}\left[O + (1-O)\frac{\mathbb{P}_f(Y=1|X)\,(1-e_m(X))}{1-\mathbb{P}_f(Y=1|X)\,e_m(X)}\right]\right] \\
&= \mathbb{E}\left[O + (1-O)\frac{\lim_{m\to\infty}\mathbb{P}_f(Y=1|X)\,(1-e_m(X))}{\lim_{m\to\infty}1-\mathbb{P}_f(Y=1|X)\,e_m(X)}\right] \\
&= \mathbb{E}\left[O + (1-O)\frac{\mathbb{P}_f(Y=1|X)\,(1-e(X))}{1-\mathbb{P}_f(Y=1|X)\,e(X)}\right] = \alpha,
\end{aligned}$$

where the first step is due to the dominated convergence theorem (the sequence of functions is bounded because of probabilities) and the second equality holds since both the factors are non zero and their limit exists for any x. □

3.2 Active PU Learning

The active learning strategy asks the user to label those examples that are the most *informative*, according to some criterion, for learning the classifier. While it is perfectly possible that strategy will query the labels for examples belonging to both classes, as discussed in the Introduction situations will arise where a user will only label positive examples. We look at both an ideal and a realistic case. In the ideal case, we assume that the user is a *perfect oracle* (subsection 3.2.1). The queried examples are labeled only if their real class is positive, and in all other cases, the user does not know the true label and the queried examples remain unlabeled. In the realistic case, we assume that the user is an *imperfect oracle* (subsection 3.2.2). The user is not always able to recognize the examples, so that with a certain probability the queried example might not be labeled. In addition, there is a low probability that the user might label a queried example as positive while its true label is negative.

Next, we describe CAPE (*Class prior estimation in Active Pu lEarning*) which is our practical approach for estimating the class prior from data. Its estimate depends on whether the user is a perfect or imperfect oracle which changes how $\overline{e_m(x)}$ is estimated. In the ideal case, the probability to label an example only depends on its true label. In the realistic case, it also depends on a probability measure that represents the user's uncertainty about its true label.

3.2.1 Propensity Scores under Perfect Oracles

The direct computation of $\overline{e_m(x)}$ breaks down into three parts. First, we compute the probability that x is labeled in a given subset with cardinality m. Second, we multiply this probability with the count of how many times x is part of a subset of size m sampled from the dataset D. Third, we compute the expected probability of sampling these subsets.

Label probability. In the ideal case, the user is a perfect oracle. If a truly positive example is selected to be labeled, the user always labels it correctly.[4] Therefore, given a subset D^* of the dataset D, an example in this subset is labeled if it is in the top-l most informative examples of D^* (denoted as D^*_l) according to the active learning strategy because those are the examples that will be queried. If we use the active learning strategy to construct a *global ranking* of all the examples in D where the higher-ranked examples are queried first, we can reasonably assume that this ranking is preserved for the examples in any subset of D. Let x_{j+1} be $j+1$-th example in the global ranking (G.R.). The probability that x_{j+1} is queried is equal to the probability that it is in the top-l of a sampled subset D^*:

$$\mathbb{P}(x_{j+1} \in D^*_l) = \begin{cases} 1 & \text{if } x_{j+1} \in \text{top-}l \text{ of G.R.} \\ \sum\limits_{t=\max\{0,m+j-N\}}^{l-1} \frac{\binom{j}{t}\cdot\binom{N-j-1}{m-t-1}}{\binom{N-1}{m-1}} & \text{otherwise.} \end{cases} \tag{3.5}$$

where t is the number of examples ranked higher in the global ranking than x_{j+1} in any given subset D^* of D.

Counting the subsets. The number of times x_{j+1} will be chosen among N elements by simultaneously selecting m examples is:

$$|\{D^* \subseteq D\colon x_{j+1} \in D^*, |D^*| = m\}| = \binom{N-1}{m-1}. \tag{3.6}$$

Expected probability of sampling the subsets. Assuming that the samples are drawn independently, the mean measure of a sample of m elements drawn from the population with distribution $p(X)$ is:

$$\mathbb{E}\left[\widehat{p}(D^*)\right] = \mathbb{E}\left[\prod_{i=1}^{m} \widehat{p}(x_i)\right] = \prod_{i=1}^{m} \mathbb{E}\left[\widehat{p}(x_i)\right], \tag{3.7}$$

[4] If a truly negative example is selected, no label is given.

where $\widehat{p}$ is estimated using a kernel density estimator. Note, we can only draw samples consisting of examples in D. Finally, the $\overline{e_m(x_{j+1})}$ for any example x_{j+1} is derived as:

$$\overline{e_m(x_{j+1})} = \mathbb{P}(x_{j+1} \in D_l^*) \times \binom{N-1}{m-1} \times \prod_{i=1}^{m} \mathbb{E}\left[\widehat{p}(x_i)\right]. \tag{3.8}$$

A final pseudo-code summarizing our approach CAPE is shown in Algorithm 1.

3.2.2 Propensity Scores under Imperfect Oracles

In the real world, the user is an imperfect oracle. If a truly positive example is selected to be labeled, she may be unsure of its label and decide not to label it. If a truly negative example is selected to be labeled, there is a small probability she mislabels it as a positive. This requires changing the label probability to include the probability that the user is able to label the example:[5]

$$\mathbb{P}(O = 1|x, D^*) = \mathbb{P}(Q = 1|x, D^*)\mathbb{P}(O = 1|x, D^*, Q = 1),$$

where Q is the binary random variable that is 1 if x is queried and 0 otherwise. Note that the previous identity is obtained because the probability of labeling an example not queried is 0.

Query probability. The probability of querying an example depends on whether or not the example is in the top-l of a subset D^* according to the active learning strategy:

$$\begin{aligned}\mathbb{P}(Q = 1|x, D^*) =& \mathbb{P}(Q = 1|D^*, x \in D_l^*)\mathbb{P}(x \in D_l^*|D^*) + \mathbb{P}(Q = 1|D^*, x \notin D_l^*)\mathbb{P}(x \notin D_l^*|D^*) \\ =& \mathbb{P}(x \in D_l^*|D^*) + \mathbb{P}(Q = 1|D^*, x \notin D_l^*)\mathbb{P}(x \notin D_l^*|D^*).\end{aligned}$$

First, if an example is in the top-l of D^*, it is always queried. Second, if an example is not in the top-l, whether it is queried now depends on the user's uncertainty about the labels of the higher-ranked examples in D^*. Let x_{j+1} be $j+1$-th example in the general ranking. To compute the probability that x_{j+1} is queried, we first simplify the problem by approximating the user's uncertainty about any example x with the mean of the user's uncertainty over all the examples in the observed dataset D. Then, the probability can be computed as:

$$\mathbb{P}(Q = 1|D^*, x_{j+1} \notin D_l^*) = \sum_{t=\max\{l,m-N+j\}}^{\min\{j,m-1\}} \binom{j}{t}\binom{N-j-1}{m-t-1} \sum_{z=t-l+1}^{t} (1-\overline{p})^z \overline{p}^{\,t-z-1} \tag{3.9}$$

[5] For brevity, we omit $|Y = 1$ everywhere in this section, even though all the events in the equations are conditioned on $Y = 1$.

where t is the number of examples ranked higher than x_{j+1} in a given subset D^*, z is the number of examples out of t that the user cannot label, and $\bar{p}$ is the user's uncertainty for any example. Intuitively, Eq. 3.9 considers all possible scenarios where the user fails to label enough examples such that x_{j+1} is queried and sums the probabilities of these scenarios.

Label probability under user's uncertainty. Finally, the label probability of an example x in position $j+1$ is:

$$\begin{aligned}\mathbb{P}(O=1|x,D^*) &= \mathbb{P}(O=1|x,D^*,Q=1)\mathbb{P}(Q=1|x,D^*)\\ &= \mathbb{P}(O=1|x,D^*,Q=1)[\mathbb{P}(x\in D_l^*|D^*)+\mathbb{P}(Q=1|D^*,x\notin D_l^*)\mathbb{P}(x\notin D_l^*|D^*)],\end{aligned} \tag{3.10}$$

where $\mathbb{P}(O=1|x,D^*,Q=1)$ is the user uncertainty of that example, $\mathbb{P}(x\in D_l^*|D^*)$ is as in Eq. 3.5, $\mathbb{P}(Q=1|D^*,x\notin D_l^*)$ is as in Eq. 3.9 and $\mathbb{P}(x\notin D_l^*|D^*)$ is $1-\mathbb{P}(x\in D_l^*|D^*)$. The final propensity score for an example x under user uncertainty is the product between the factors in Eq. 3.6, 3.7, and 3.10.

Algorithm 1 Pseudo-code for the algorithm of CAPe.

Input: A PU dataset $D=\{(x,o)\}$ of size N; a query budget l; an AL strategy Γ assigning uncertainty scores (higher = more likely to be selected).
Output: α, the class prior.

$f \leftarrow$ Fit a non-traditional classifier on D
$\pi \leftarrow$ Compute_probability(f, D) // $\mathbb{P}_f(Y=1|X=x)$
GR $\leftarrow$ Sort$(\{\Gamma(x)\colon x\in D\}$, order ="descending")
$e_m \leftarrow$ Array(N) // initialize prop. scores
$m_{\text{LIST}} \leftarrow \{0.02, 0.04, 0.06, \ldots, 0.4, 0.5, \ldots, 0.9\}\cdot N$
for m^* in m_{LIST} **do**
    $D^* \leftarrow$ Sample$(D,$ size $= m^*)$
    $\overline{e_m} \leftarrow$ Array(N)
    for $j \in [1,\ldots,N]$ **do**
        $\overline{e_m}[j] \leftarrow$ Compute_propensity$(x_j, D^*, l, \text{GR}, N)$ // see Eq. 3.8
        $e_m[j] \mathrel{+}= \overline{e_m[j]}$
    end for
end for
$e \leftarrow e_m/|m_{\text{LIST}}|$
$\alpha \leftarrow$ Compute_prior(e, π, D) // see Eq. 3.1
return α

3.3 Experiments

We address the following empirical questions:

Q1. Can CAPE accurately estimate the true class prior α?

Q2. How does user uncertainty affect CAPE's ability to estimate the class prior?

Q3. Does a more accurate estimate of the class prior improve the performance of an anomaly detector?[6]

3.3.1 Experimental Setup

Methods. We compare: our proposed method CAPE[7], TIcE which estimates the class prior using decision tree induction [19], and KM1 and KM2 which compute the class prior by modeling the distribution of the positive examples [194].

Data. The benchmark consists of 9 standard anomaly detection datasets from [33].[8] The datasets are listed in Table 3.1. They contain more normals than anomalies with the normal class prior α varying between 0.64 and 0.99.

Dataset	# Examples (N)	# Vars (d)	α	γ
WBC	454	9	0.9780	0.0220
Shuttle	507	9	0.9862	0.0138
WDBC	367	30	0.9728	0.0272
Stamps	340	9	0.9088	0.0912
Ionosphere	351	32	0.6410	0.3590
Cardiotocography	434	21	0.9493	0.0507
PageBlocks	421	10	0.8979	0.1021
Pima	625	8	0.8000	0.2000
Annthyroid	713	21	0.9257	0.0743

Table 3.1: Benchmark anomaly detection datasets (γ is the contamination).

[6] The anomaly detection algorithms use the *contamination factor* $\gamma = 1 - \alpha$ to transform scores into binary predictions.

[7] Code: `https://github.com/Lorenzo-Perini/Active_PU_Learning`

[8] Data: `www.dbs.ifi.lmu.de/research/outlier-evaluation`

Setup. In all experiments, SSDO with its default parameters is used as the semi-supervised anomaly detector [237].[9] SSDO learns a classifier from unlabeled and normal examples. We use IF [150] as its unsupervised prior. Using the method from [129], the anomaly scores are mapped to probabilities. We use *uncertainty sampling* as active learning strategy [211]. We model the user's uncertainty using the kernel density estimate as implemented in Scikit-Learn version 1.4.0. For each dataset and compared method, the following procedure is repeated five times. First, the dataset is split into training and test sets using a stratified 5-fold split. All training data are initially unlabeled. Then iteratively, the user is queried until $l = 5$ new labels are added to the training set following the active learning strategy, and the probability of labeling the example correctly is equal to the user's uncertainty. After adding new labels to the training data, the class prior is estimated on the training data, the SSDO classifier is retrained, and its performance on the test set is measured (using the estimated class prior to obtaining binary predictions for the test data). The process stops when 150 examples are labeled. We report the results averaged over all five runs.

Hyperparameters. The parameters of TIcE, Km1, and Km2 are set to the values recommended in the original papers [220]. CAPe has only one hyper-parameter: the range of cardinalities m in the outer loop, which we set to $N \cdot \{0.02, 0.04, 0.06, \ldots, 0.4, 0.5, \ldots 0.9\}$.

3.3.2 Experimental Results

Q1: Recovering the class prior. Figure 3.1 shows the mean absolute error (*MAE*) of the estimated class prior as a function of the number of labeled training examples with no user uncertainty. On seven of the nine datasets, CAPe outperforms the three baselines. On these datasets, our estimate converges to the correct one, often with < 100 labels. On two datasets, TIcE results in (slightly) better performance than CAPe. In these two datasets, our approach tends to overestimate the class prior, likely due to inaccuracies in the underlying SSDO model (i.e., anomalous examples have a high predicted probability of being normal). In addition, as more examples are labeled, CAPe's estimate of the class prior converges to the true class prior smoothly. In contrast, acquiring a small number of labels (e.g., 5) may cause its competitors' estimates of the class prior to change dramatically.

Q2: Impact of user uncertainty. We repeat the previous experiment in a more realistic setting where the user is uncertain and makes mistakes in the labeling. Figure 3.2 shows how the mean absolute error (*MAE*) of the estimated class prior

[9] Code: `https://github.com/Vincent-Vercruyssen/anomatools`

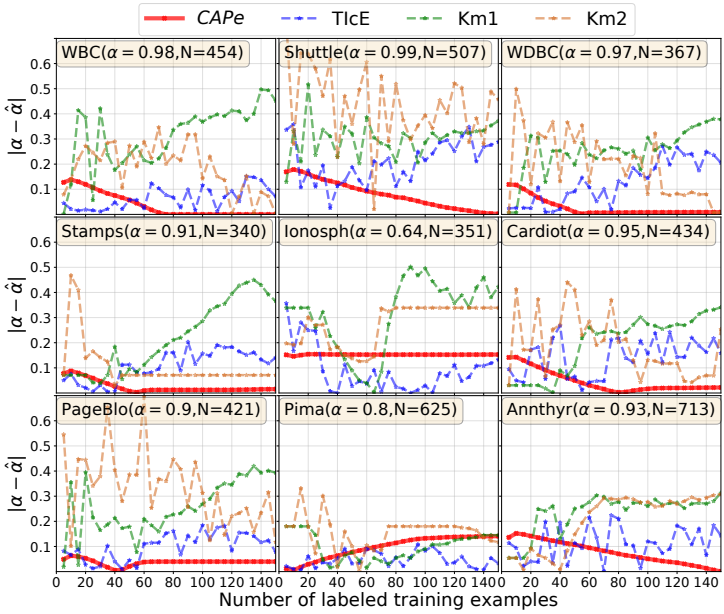

Figure 3.1: MAE of class prior estimates as a function of the number of labels, under no user uncertainty. Lower numbers are better.

varies as a function of the number of labeled instances for each method. Again, CAPE results in the most accurate estimates on seven of the nine datasets. Again, its estimates fluctuate less than its competitors.

Q3: Class prior impact on anomaly detection. Most anomaly detectors require knowing the proportion of anomalies in the dataset either for training the detector itself or for thresholding the detector's numeric outlier scores to be able to decide practice. In this experiment, we consider the second scenario and use the estimated class prior to converting SSDO's numeric output into a decision rule. Figure 3.3 shows the F_1 score for SSDO's model when using the class prior estimated by each method to set the decision threshold. Here, the results are more mixed as CAPE yields equivalent or more accurate estimates in a small majority of the cases. Note that the black dashed line represents performance when using the true class prior: using the true and estimated priors result in similar predictive performance.

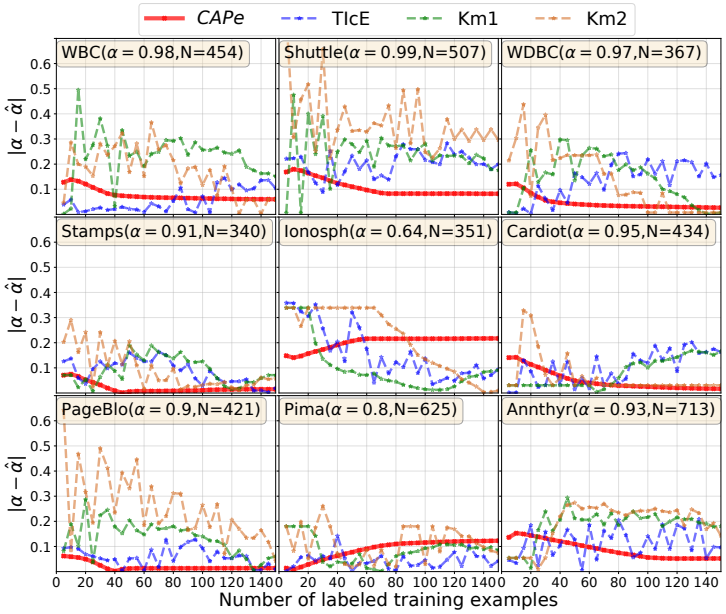

Figure 3.2: MAE of class prior estimates as a function of the number of labels, assuming the user's uncertainty. Lower numbers are better.

3.4 Related Work

Several papers have studied the combined setting of PU learning and active learning. However, while we focus on estimating the class prior, these papers have a different focus. Ghasemi et al. [79] designed an uncertainty sampling active learning strategy specifically tailored to PU datasets. Barnabé-Lortie et al. [17] developed an active learning strategy for one-class classification by querying the examples that match the learned class the least. There are a number of different ways to apply active learning strategies when dealing with one-class classification problems [228, 229]. Finally, G. He et al. [100] applied active learning to PU time series data by querying the examples with both high uncertainty and high utility.

3.5 Conclusion

In summary, this Chapter introduced a novel approach CAPe for estimating the class prior in a Positive-Unlabeled (PU) scenario, where positive labels (i.e., normals) are

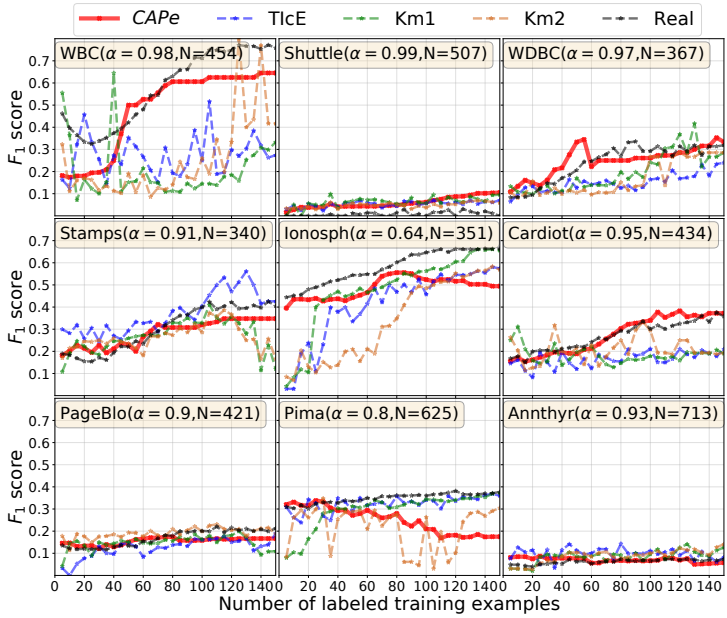

Figure 3.3: The F_1 score when using each approach's estimated class prior to threshold SSDO's numeric output into a decision rule.

collected through an active learning strategy. Our method computes the class prior by estimating the propensity score for each unlabeled example, which represents how likely the active learning strategy is to query that example's label to a user. Our theoretical analysis proves that CAPE's class prior estimates converge to the true value, along with the estimated propensity scores. We analyzed two settings for estimating the propensity scores: one assuming the user makes no mistakes and labels only positive examples (oracle), and the other considering that the user may make mistakes (imperfect user). Experimentally, we showed that CAPE outperforms existing approaches in (1) achieving accurate estimates of the class prior and (2) increasing an anomaly detector's performance by leveraging the estimated class prior to transform scores into hard predictions.

Limitation. Our theoretical analysis shows that CAPE's estimates converge when sampling infinitely many subsets, which cannot be done in a real-world setting. Increasing the number of drawn subsets yields better estimates but enlarges the computational effort. Thus, future work would target improving the computational cost

of CAPE by, for example, finding alternatives to sampling D. [10]

In the next Chapter, we investigate another realistic setting for anomaly detection: monitoring multiple related assets, such as a fleet of wind turbines. Specifically, we study how to leverage the contamination factor, which is given for one (source) dataset, to estimate the contamination of another (target) dataset.

[10] Currently, we just provide a fast version of CAPE implemented in Cython.

Chapter 4

Transferring the contamination factor between anomaly detection domains by shape similarity

Real-world anomaly detection tasks often involve monitoring a *fleet of related entities* such as machines [197], windmill farms [255] or retail stores [237]. While the entities' behaviors are related in such cases, there are important differences that will affect the collected data. For instance, windmill-specific properties (e.g., orientation, size, location) or store-specific properties (e.g, size, services, or opening hours) will affect the observed data. Consequently, how many anomalies are present will vary from entity to entity. Given that such tasks may involve monitoring 100s of entities, estimating the contamination factor for each one separately by labeling data would be too onerous. Thus, an interesting avenue to explore is whether it is possible to *transfer* a known contamination factor from data about one entity (the source domain) to data collected from another similar entity (the target domain). If this were possible, it would significantly decrease the labeling burden, as one would no longer need to collect labels for all entities.

Problem Statement

The problem that this chapter addresses is the following:

Given: An unlabeled source dataset D^S with a known contamination factor γ^S, an unlabeled target dataset D^T_m, and an anomaly detector f;

Do: Estimate the contamination factor γ^T_m of the target domain.

Contributions of this Chapter

This chapter proposes TrADe (*transferring the contamination factor between anomaly detection domains by shape similarity*), the first algorithm for transferring the known contamination factor from a source domain to a target domain where it is unknown. TrADe's key assumption is that if the distributions over the anomaly scores of the normal examples computed by a given anomaly detection algorithm, are similar in shape in both the source and target domain, the target anomaly score threshold can be derived from the (known) source threshold. First, we use the known source contamination factor to construct a proper distribution over the normal examples in the source domain. Then, we find a threshold on the target domain anomaly scores that makes the distribution over the anomaly scores of the resulting "normal" target examples as similar as possible to the earlier-derived source distribution. This is constructed as an optimization problem. Finally, we use the resulting threshold to infer the target domain's contamination factor. We theoretically analyze our approach and prove that the estimated target contamination factor converges to its true value when the distribution of the target scores becomes closer to their ground-truth distribution. Empirically, we performed an extensive evaluation on 206 source-target pairs arising from three real-world domains: detecting anomalous water usage in retail stores, detecting blade icing on windmills, and detecting botnets on IoT traffic data. We find that TrADe outperforms multiple competitors.

The content of this chapter is based on the following publication [188]:[1]

Perini, L., Vercruyssen, V., and Davis, J. Transferring the Contamination Factor between Anomaly Detection Domains by Shape Similarity. In *Proceedings of the Thirty-Six AAAI Conference on Artificial Intelligence (AAAI 2022)*, volume 36, pp. 4128–4136.

4.1 Methodology

Our method TrADe estimates γ^T_m as follows. First, TrADe trains two separate anomaly detectors. It trains one on the source data and uses it to assign an anomaly score to each example in D^S. It trains the other one on the target data and uses it to assign an anomaly

[1] LP provided the main body of the work (code, theory, text), VV assisted with experiments and nailing down the right research questions, JD guided formalizing and structuring the text.

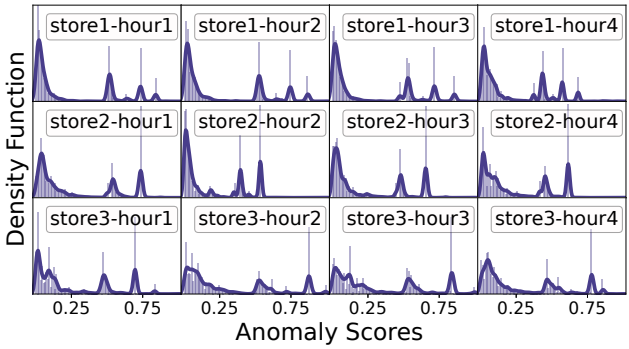

Figure 4.1: Illustration of how the distribution anomaly scores produced by the same anomaly detection algorithm f on related real-world stores exhibit a similar shape.

score to each example in D^T. Because the domains are related and normal behaviors are similar, the key insight is that the source and target distributions of the normal examples' anomaly scores will be **similar** (but not necessarily equal). That is, there may be scales, offsets or shifts but not fundamental changes in the underlying distribution. Figure 4.1 motivates this assumption, showing that anomaly scores produced by algorithm f on multiple related domains follow a similar distribution when looking only at the low scores, which by construction correspond to the normal examples. However, because both datasets are unlabeled, we do not know the distribution of the normal examples' anomaly scores. Second, TrADe uses the known source contamination factor γ^S to set a threshold λ^S on the source anomaly scores. Examples with an anomaly score lower than λ^S are considered normal, yielding the distribution over their anomaly scores, which we call the λcut distribution. Third, TrADe derives the target threshold λ_m^T by solving an optimization problem: λ_m^T is chosen such that shapes of the resulting λ_m^Tcut distribution and the λ^Scut distribution are as **similar** as possible. This leverages our earlier insight. Finally, TrADe predicts the target contamination factor γ_m^T as the proportion of target examples with an anomaly score above the value of λ_m^T. The following subsections describe each of these steps in detail. Then, we explore the theoretical properties of TrADe.

Notation. In this chapter, we focus on the *anomaly score random variables* S, T, and T_m referring to the ground-truth anomaly scores of, respectively, the source domain, the target domain, and the sampled target dataset. We linearly normalize their distributions to have support in $[0, 1]$ and denote by s, t, and t_m their probability density functions.[2] Finally, we denote their contamination factors by $\gamma^S = \mathbb{P}\left(Y^S = 1\right)$,

[2]We use this notation instead of, respectively, $p(s)$, $p(t)$ and $p(t_m)$ because in this chapter we do not provide any point-wise analysis but we only deal with distributions and densities.

$\gamma^T = \mathbb{P}\left(Y^T = 1\right)$, $\gamma_m^T = \mathbb{P}\left(Y_m^T = 1\right)$, where 1 is the anomaly class. Because our analysis requires highlighting the number of examples belonging to the target dataset, every target domain variable is indicated with the index m, such as the dataset D_m^T (of size $|D_m^T| = m$), which is a small (and therefore potentially non-representative) set of examples drawn from $p(X^T, Y^T)$.

4.1.1 Modeling the Distribution of the Anomaly Scores of the Normal Examples in D^S

Modeling the distribution of anomaly scores assigned to the source normal examples is challenging because we lack labels. Instead, we exploit the fact that the source domain's contamination factor γ^S is known. First, we set a threshold λ^S on the source anomaly scores such that the proportion of examples with score $> \lambda^S$ is equal to γ^S. Then, we model the distribution of normal scores as the distribution of scores $\leq \lambda^S$ by performing a normalization such that the support of the new distribution is again $[0, 1]$ and its area is equal to 1. More generally, for an arbitrary threshold value λ, we call this derived distribution the *λcut distribution* and define it as follows:

Definition 4. *Let U be any random variable with support in $[0, 1]$ and probability density function u. Then, for any $\lambda \in [0, 1]$, we define the* λcut distribution *of U as:*

$$u^\lambda(x) \coloneqq u(\lambda x) \cdot \frac{\lambda}{\int_0^\lambda u(x')\,dx'} \qquad \text{for any } x \in [0, 1].$$

Proposition 3. *For any $\lambda \in [0, 1]$, u^λ is a probability density function.*

Proof. See the Appendix A for the formal proof. □

This step assumes that the anomaly detection algorithm yields a reasonable ranking of the examples from least to most anomalous. However, even if the ranking is not perfect, the subsequent transfer step can still be accurate because the same algorithm is used to derive both the source and target λcut distributions. Thus, incorrect predictions are likely similarly distributed in both domains.

4.1.2 Finding the Target Threshold λ_m^T via Transfer

If we knew the threshold λ_m^T on the target anomaly scores that separates the normal examples from the anomalies, we could trivially estimate the target contamination factor. Therefore, we attempt to derive λ_m^T by exploiting our assumption that the source and target distributions of the normal examples' anomaly scores are similar (given they

are derived using the same anomaly detector). This can be solved by attempting to find a value λ_m^T that yields a λ_m^Tcut distribution in the target domain that is similar to the source's λ^S cut distribution. We can measure the similarity between two distributions $p(S)$ and $p(T)$ using the Kullback-Leibler (KL) divergence:

$$KL(S \parallel T) = \int_0^1 s(x) \log\left(\frac{s(x)}{t(x)}\right) dx,$$

where s and t are continuous probability density functions. Intuitively, the KL divergence quantifies the amount of information lost when approximating S with T with small KL divergence scores corresponding to little lost information, and hence similar shapes. We selected the KL divergence for three reasons. First, its theoretical properties enable a convergence study [75]. Second, it is a widely used measure in the literature [24]. Third, it is stronger than several other similarity measures (e.g., maximum gap) as they are upper bounds of KL [80].

We formulate our task as finding the threshold λ_m^T such that the KL divergence between the corresponding target λ_m^Tcut distribution and the source's λ^Scut distribution is minimal:

$$\lambda_m^T = \underset{\lambda \in [\delta, 1]}{\arg\min} \left\{ KL\left(S^{\lambda^S} \;\middle\|\; T_m^{\lambda}\right) \right\}, \tag{4.1}$$

where S^{λ^S} and T_m^{λ} are the random variables with, respectively, the λ^Scut and λcut distributions. The $\delta > 0$ is a small value that depends on the detector f and on the datasets, and represents the lower boundary for the choice of λ_m^T. The contamination factor is usually small such that $\lambda_m^T > 0$. If $\lambda_m^T = 0$, all the examples would be anomalous.

Theoretically, there may be more than one solution to Equation 4.1 because the objective might not be smooth such that arg min returns a set of solutions. However, in practice, this is unlikely to occur and it did not happen in our experiments.

4.1.3 Deriving the Target Contamination Factor

Mirroring the reasoning for setting the source threshold λ^S, a reasonable estimate of the target domain's contamination factor can be derived by looking at the proportion of examples in the target domain with an anomaly score greater than λ_m^T. Theoretically, given the target threshold $\lambda_m^T \in [\delta, 1]$ we should estimate the contamination factor through the continuous score variable T_m as $\mathbb{P}\left(f\left(X^T\right) \geq \lambda_m^T\right) = \mathbb{P}\left(T_m \geq \lambda_m^T\right)$. However, because in practice we can only use a finite number of examples, we estimate the contamination factor as the discrete proportion of examples with anomaly scores

greater than λ_m^T:

$$\hat{\gamma}_m^T := \frac{\left|\left\{f(x) \geq \lambda_m^T \,\middle|\, x \in D_m^T\right\}\right|}{m} = \frac{\sum_{i=1}^{m} \mathbb{1}_{\{f(x)\geq\lambda_m^T\}}(x_i)}{m} \tag{4.2}$$

where $|\cdot|$ indicates the cardinality of a set, $\mathbb{1}$ is the indicator function, $f(x)$ is the anomaly score of the example $x \in D_m^T$ and λ_m^T is the transferred target predictive threshold. In the following proposition, we prove that if the target threshold λ_m^T is correct, our estimator $\hat{\gamma}_m^T$ is unbiased, meaning that it recovers the target domain's true contamination factor γ_m^T.

Proposition 4. *Given the target threshold* $\lambda_m^T \in [\delta, 1]$ *such that* $\mathbb{P}\left(Y_m^T = 1\right) = \mathbb{P}\left(T_m \geq \lambda_m^T\right)$*, the contamination factor's estimator defined in Eq. 4.2 is unbiased.*

Proof. See the Appendix A for the formal proof. □

4.1.4 Choice of Anomaly Detection Algorithm f

In theory, TrADe can use any anomaly detection algorithm f to estimate $\hat{\gamma}_m^T$. In practice, we find that using an ensemble of anomaly detectors yields better results.[3] First, each detector i in the ensemble produces an estimate of the target contamination factor as described above. Then, TrADe computes the final estimate $\hat{\gamma}_m^T$ as a weighted average of each ensemble member's estimate. The weight of each member w_i is inversely proportional to its obtained KL divergence KL_i:

$$w_i = \frac{1}{|E| - 1} \times \left(1 - \frac{KL_i(S, T_m)}{\sum_{j=1}^{|E|} KL_j(S, T_m)}\right), \tag{4.3}$$

where $|E|$ is the number of detectors in the ensemble. This weighting scheme awards ensemble members that produce similar score distributions for the source and target domain.

A final pseudo-code summarizing our approach TrADe is shown in Algorithm 2.

4.2 Theoretical Convergence Analysis

Our main theoretical result is Theorem 5, which states that our approach for estimating the contamination factor will converge to the theoretical target value in the limit. This theorem rests on making the following two theoretical assumptions.

[3] We provide empirical evidence in the experimental section.

Algorithm 2 Pseudo-code for the algorithm of TrADe.

Input: A source dataset D^S of size N with contamination γ^S; a target dataset D_m^T of size m; a set of anomaly detectors $E = \{f_1, \dots f_{|E|}\}$.
Output: $\hat{\gamma}_m^T$, the target contamination factor.

for f_i in E **do**
  $s_i \leftarrow f_i\left(D^S\right)$
  $s_i^{\text{CUT}} \leftarrow$ Remove values above the $(1-\gamma^S)$-th percentile from s_i
  $s_i^{\text{CUT}} \leftarrow$ Min-max normalize(s_i^{CUT})
  $t_i \leftarrow f_i\left(D_m^T\right)$
  $\lambda_m^T \leftarrow \text{Minimize}_{\lambda\in[0,1]}\left(\text{Compute\_KL}(s_i^{\text{CUT}}, t_i, \lambda)\right)$ // see Eq. 4.1
  $\gamma_{m,i}^T \leftarrow |\{t_i \geq \lambda_m^T\}|/m$ // see Eq. 4.2
  $KL_i \leftarrow \text{Compute\_KL}(s_i, t_i, \max t_i)$
end for
for f_i in E **do**
  $w_i \leftarrow \frac{1}{|E|-1} \times \left(1 - \frac{KL_i}{\sum_{j=1}^{|E|} KL_j}\right)$ // see Eq. 4.3
end for
$\hat{\gamma}_m^T \leftarrow \sum_i^{|E|} w_i \cdot \gamma_{m,i}^T$ // see Eq. 4.2
return $\hat{\gamma}_m^T$

Algorithm 3 Compute_KL(s_i, t_i, λ): Compute KL for a λcut distribution.

Input: Training scores s_i and target scores t_i; a threshold λ.
Output: KL(S, T_m^λ).

$t_i^{\text{CUT}} \leftarrow$ Remove values above λ from t_i
$t_i^{\text{CUT}} \leftarrow$ Min-max normalize(t_i^{CUT})
$p_S \leftarrow \text{KDE}(s_i)$ // fit source kernel density estimator
$p_T \leftarrow \text{KDE}(t_i^{\text{CUT}})$ // fit target kernel density estimator
KL$(S, T_m^\lambda) \leftarrow \int_0^1 p_S(x) \log(p_S(x)/p_T(x))\, dx$
return KL(S, T_m^λ)

Assumption 1. We assume that the score examples of the source domain are an i.i.d. drawn from the real distribution $p(S)$. This is coherent with a practical setting, where the source dataset is large enough to represent the ground truth distribution. On the other hand, we assume that there may be some bias in the distribution of scores $p(T_m)$ with respect to $p(T)$, and that the bias gradually fades out when adding examples. Formally, we require that, for $m \to +\infty$, $t_m \to t$ **uniformly** in $[0, 1]$, which means that, for every $\varepsilon > 0$, there exists $M \in \mathbb{N}$ such that, for all $m \geq M$ and $x \in [0, 1]$, the inequality $|t(x) - t_m(x)| < \varepsilon$ holds. We also indicate this assumption by $T_m \to T$.

Assumption 2. We assume that the normal scores distribution of the theoretical target distribution $p(T)$ shares exactly the same shape as the normal scores distribution of the source domain. Formally, we require that $KL(S^{\lambda^S} \parallel T^{\lambda^T}) = 0$, where S^{λ^S} and T^{λ^T} represent the random variables of the two domains' normal scores. This assumption is a theoretical generalization of what Figure 4.1 shows.

Formally, our **main theoretical result** is stated as:

Theorem 5. *Let S and T_m be two continuous random variables representing the anomaly scores produced by an anomaly detector f on, respectively, the source (D^S) and the target (D_m^T) domains. Assume that T is the random variable with the ground-truth distribution of the target domain scores. Let γ^S be the contamination factor of the source domain. Let us fix $\delta > 0$ small enough and let λ^S and λ^T be the real predictive thresholds of S and T. Let's assume that s, t and t_m are the positive densities of S, T and T_m such that $t_m \to t$ uniformly in $[0, 1]$ (Assumption 1) for $m \to +\infty$ and that $KL(S^{\lambda^S} \parallel T^{\lambda^T}) = 0$ (Assumption 2). Also, let $\lambda_m^T \in [\delta, 1]$ be the estimate of the target predictive threshold through Eq. 4.1. Then,*

$$\lim_{m \to +\infty} \lambda_m^T = \lambda^T.$$

Furthermore, let $\hat{\gamma}_m^T$ be the estimate of the target contamination factor by the estimator defined in Eq. 4.2. Then,

$$\mathbb{E}\left[\hat{\gamma}_m^T\right] \to \gamma^T \quad \text{for } m \to +\infty.$$

Proof. We now sketch the proof for this theorem. The detailed proofs are in Appendix A along with the supporting theorems used in the sketch. In order to prove the first part, we need to motivate the transition of the limit symbol through the functions, following these steps:

$$\lambda^T \overset{\text{(i)}}{=} \arg\min_{\lambda \in [\delta,1]} \left\{ KL\left(S^{\lambda^S} \middle\| T^{\lambda}\right) \right\} \overset{\text{(ii)}}{=} \arg\min_{\lambda \in [\delta,1]} \left\{ KL\left(S^{\lambda^S} \middle\| \lim_{m \to +\infty} T_m^{\lambda}\right) \right\}$$

$$\overset{\text{(iii)}}{=} \arg\min_{\lambda \in [\delta,1]} \left\{ \lim_{m \to +\infty} KL\left(S^{\lambda^S} \middle\| T_m^{\lambda}\right) \right\} \overset{\text{(iv)}}{=} \limsup_{m \to +\infty} \arg\min_{\lambda \in [\delta,1]} \left\{ KL\left(S^{\lambda^S} \middle\| T_m^{\lambda}\right) \right\} \overset{\text{(v)}}{=} \lim_{m \to +\infty} \lambda_m^T. \tag{4.4}$$

The first (i) and the last (v) equalities come from the uniqueness of the solution shown in Theorems 13 and 14; the second step (ii) is motivated by the convergence of λcut distributions proved in Theorem 15; the third equality (iii) holds by Theorem 16; the fourth result (iv) is guaranteed by Theorems 17 and 18. Note that the equal in (iv) is not an inclusion because of the uniqueness of the solution λ^T (shown in Theorem 14).

Once we proved that the threshold converges as expected, the second part of this theorem focuses on the contamination factor's convergence, which comes directly as follows:

$$\lim_{m\to+\infty}\mathbb{E}\left[\hat{\gamma}_m^T\right] \overset{\text{(i)}}{=} \lim_{m\to+\infty}\mathbb{E}\left[\frac{\sum_{i=1}^m \mathbb{1}_{\{f(x)\geq\lambda_m^T\}}(x_i)}{m}\right] \overset{\text{(ii)}}{=} \lim_{m\to+\infty}\frac{\sum_{i=1}^m \mathbb{E}\left[\mathbb{1}_{\{f(x)\geq\lambda_m^T\}}(x_i)\right]}{m}$$

$$\overset{\text{(iii)}}{=} \lim_{m\to+\infty}\frac{\sum_{i=1}^m \mathbb{E}\left[\mathbb{1}_{\{T_m\geq\lambda_m^T\}}\right]}{m} \overset{\text{(iv)}}{=} \mathbb{E}\left[\lim_{m\to+\infty}\mathbb{1}_{\{T_m\geq\lambda_m^T\}}\right] \overset{\text{(v)}}{=} \mathbb{E}\left[\mathbb{1}_{\{T\geq\lambda^T\}}\right] \overset{\text{(vi)}}{=} \mathbb{P}\left(T\geq\lambda^T\right) = \gamma^T$$

The first equality (i) holds by our definition of the estimator (Eq. 4.2); the second step (ii) exploits the properties of the expectation; the third equality (iii) follows from the fact that x_i is i.i.d.; the interchange between the expectation and the limit (iv) is allowed by the theorem of dominated convergence; the result of the limit (v) is motivated by both the assumptions of uniform convergence ($T_m \to T$) and the first part of this theorem ($\lambda_m^T \to \lambda^T$); finally, the last step (vi) is a property of the characteristic function. □

4.3 Experiments

Using the pseudo-code in Algorithm 2, we address the following empirical questions:

Q1. Does TrADe accurately estimate the true target contamination factor?

Q2. Does a more accurate estimate of the target contamination factor improve the performance of the anomaly detector?

Q3. Does an ensemble of anomaly detectors produce a more accurate estimate of the target contamination factor than a single detector f?

Q4. How does TrADe perform when varying the source contamination factor?

4.3.1 Experimental Setup

Methods. We compare TrADe[4] against five baselines. Source$_\gamma$ simply assumes the target contamination factor to be equal to the source contamination factor. Source$_\lambda$ first uses an ensemble to estimate λ_m^T through a simple average of the ensemble members' estimates. Then, it estimates the target contamination factor as the proportion of target examples with anomaly score $> \lambda_m^T$. Coral [225] is an unsupervised domain adaptation technique that transforms the source distribution to be similar to the target distribution. After applying this transformation, it uses Source$_\lambda$ approach to estimate the target contamination factor. Finally, Unify [129] and Otsu [175] are unsupervised approaches that can be applied to the target anomaly scores. The former transforms the anomaly scores into posterior probabilities and estimates the contamination factor as the proportion of target examples with posterior anomaly probability > 0.5. The latter selects the best-separating threshold by minimizing the intra-class variance and estimates the contamination factor as the proportion of scores above the threshold.

Data. Our experiments focus on how anomaly detection can impact real-world sustainability and security. Specifically, we look at preventing water loss, preventing blade icing in wind turbines, and detecting IoT traffic anomalies. For the first task, we use 12 proprietary water consumption datasets obtained in collaboration with a large retail company.[5] Each dataset contains the water consumption measured each day during a particular hour-long segment in one of three retail stores over the course of 4.5-5 years. The measurement interval is 5 minutes. The raw consumption data of each hour-long segment are transformed into feature-vectors.[6] The goal is to detect hours of anomalous consumption (e.g., a leak). Accurate detection of the anomalies aids the company in preventing water losses, which can otherwise easily amount to 1000s of litres a year. For the second task, we use two public wind turbine datasets [254]. Various measurements (e.g., wind speed, power, etc.) are collected approximately every 7 seconds for either two months (turbine 15) or one month (turbine 21). We construct feature-vectors from the data as in the original paper, averaging over time segments of 1 hour. The goal is to detect ice formation on wind turbine blades, which could potentially damage the turbines and slow power production. To obtain the wind turbine data, see the original paper [254]. For the third task, we use 9 public[7] IoT datasets [162, 164]. Each dataset contains real traffic data, collected from one commercial IoT device infected by authentic botnets in an isolated network. The features include statistics on the stream data (e.g., source IP, MAC, channel jitter, socket), time-frame (e.g, the decay

[4] `https://github.com/Lorenzo-Perini/TransferContamination`

[5] The data was provided under an NDA and cannot be shared.

[6] We use 9 statistical (average, standard deviation, max, min, median, sum, entropy, skewness, curtosis) and 2 binary features (whether its Friday or Sunday), 11 in total.

[7] `https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT#`

factor), and statistics extracted from the packet stream (e.g., weight, mean, std, radius, magnitude) for a total of 115 attributes. For computational reasons, we use a random subsample of 2000 examples for each dataset. The Appendix A contains additional details.

Setup. Each experiment goes as follows: (i) pick a source-target dataset pair from the benchmark; (ii) train a separate anomaly detector on both the source and target domains and use them to compute the anomaly scores; (iii) estimate the target contamination factor and use it to make the target anomaly predictions; (iv) evaluate the estimated contamination factor using the *mean absolute error* (MAE) and the predictions using the F_1 *score*; and (v) derive the average relative improvements:

$$\text{MAE improvement} = \frac{\text{MAE}_{\text{BASELINE}} - \text{MAE}_{\text{TRADE}}}{\text{MAE}_{\text{BASELINE}}};$$

$$\text{F}_1 \text{ improvement} = \frac{\text{F}_{1\,\text{TRADE}} - \text{F}_{1\,\text{BASELINE}}}{\text{F}_{1\,\text{BASELINE}}}.$$

In step (i) we do not mix the three types of datasets, as it would violate Assumption 2. For the water and wind turbines tasks, each dataset serves once as the target domain while the remaining ones serve as a source domain yielding $12 \times 11 + 2 \times 1 = 134$ source-target pairs. For the IoT data, before taking a subsample we set the target contamination factor to 0.01 and vary the source contamination factor in $[0.03, 0.05, 0.08, 0.10, 0.15, 0.20, 0.25]$. This results in $9 \times 8 = 72$ source-target pairs.

Hyperparameters. TRADE, SOURCE$_\lambda$, CORAL, UNIFY, and OTSU all use an ensemble of 9 unsupervised anomaly detectors from different families (distance-based, statistical, and spectral): KNNO [195], CBLOF [101], HBOS [82], SOD [130], IF [150], COPOD [145], LODA [189], LSCP [257] with three LOF [29], and VAE [31]. Their hyperparameters are set to the default values [220].[8]

TRADE uses *differential evolution* [222] (maxit. = 100, mut. = 0.4, rec. = 0.2) as the optimization solver. We restrict the solution to be in the interval $(0, 0.25)$.

Computational cost. The most expensive step of TRADE is the optimization algorithm. For a single experiment, the CPU time is $\sim$ 10000 seconds. To run all experiments, we use an internal cluster of six 24- or 32-thread machines (128 GB of memory). The experiments finish in $\sim$ 24 hours.

[8] See Appendix A for details.

4.3.2 Experimental Results

Error on γ	**MAE**	**Ranking**	**# times TrADe**		
Method	**Avg. ± SD**	**Avg. ± SD**	**W**	**D**	**L**
TRADE	**0.060 ± 0.035**	**2.14 ± 1.04**	-	-	-
SOURCE$_\gamma$	0.075 ± 0.042	2.90 ± 1.49	**139**	4	63
SOURCE$_\lambda$	0.112 ± 0.080	3.98 ± 1.60	**158**	6	42
CORAL	0.114 ± 0.072	4.22 ± 1.54	**173**	1	32
UNIFY	0.095 ± 0.044	3.53 ± 1.55	**157**	2	47
OTSU	0.137 ± 0.078	4.23 ± 1.86	**159**	2	45
Performance	F_1 ***score***	**Ranking**	**# times TrADe**		
Method	**Avg. ± SD**	**Avg. ± SD**	**W**	**D**	**L**
TRADE	**0.32 ± 0.21**	**2.72 ± 1.36**	-	-	-
SOURCE$_\gamma$	0.29 ± 0.21	3.03 ± 1.58	**994**	114	746
SOURCE$_\lambda$	0.27 ± 0.19	3.86 ± 1.55	**1249**	125	480
CORAL	0.27 ± 0.21	3.91 ± 1.53	**1241**	158	455
UNIFY	0.27 ± 0.19	3.87 ± 1.65	**1205**	143	506
OTSU	0.26 ± 0.14	3.61 ± 1.98	**1154**	10	690

Table 4.1: Comparison of TRADE with the baselines. The top part of the table shows the average MAE of each method's estimate of the target contamination factor, the average MAE rank ± standard deviation (SD) of each method, and the number of times TRADE wins (lower MAE), draws, and loses (higher MAE) against each baseline (absolute differences ≤ 0.001 count as draw). The bottom part of the table shows similar information for the F_1 score, averaged over the 9 considered detectors.

Q1. Estimating the target contamination factor γ_m^T. Table 4.1 (top) summarizes the results of using TRADE and the baselines to estimate the target contamination factor in each of the 206 source-target pairs. TRADE obtains the lowest (best) average MAE rank (computed following [51]). On average, it achieves the lowest MAE of the target contamination factor's estimate across all experiments. TRADE estimates γ_m^T with a lower/similar error than each baseline in at least $\sim 69.5\%$ of the experiments.

Figure 4.2 (left) shows TRADE's average improvement in MAE compared to the baselines aggregated for each of the 23 target domains. Positive values imply that TRADE achieves a lower, i.e., better, MAE. TRADE produces better average estimates of the target contamination factor on 13 vs. SOURCE$_\gamma$, 17 target domains vs. UNIFY, 21 vs. CORAL, SOURCE$_\lambda$ and OTSU.

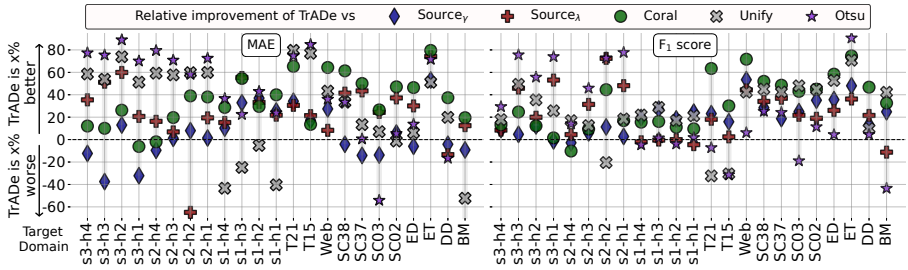

Figure 4.2: Average relative improvement in MAE (left) and the F_1 (right) of TrADe versus each baseline, aggregated per target domain (x-axis). Positive values indicate that TrADe performs better than the baseline. For each target domain, TrADe's relative improvement in MAE varies between 15% (vs Source$_\gamma$) and 40% (vs Coral), while the F_1 score improves by at least 22% (vs Source$_\gamma$) and up to 35% (vs Coral).

We perform the Friedman rank test to test the null-hypothesis that all compared methods perform similarly [51, 112]. The obtained Friedman corrected statistic of 59 and corresponding p-value of $\approx 10^{-16}$ allow us to reject this null-hypothesis. Applying the Bonferroni-Dunn post-hoc test [59] with $\alpha = 5$, shows that TrADe's performance is statistically significantly better than all the baselines.

Q2. Impact of estimating the target contamination factor correctly on the performance of the anomaly detector. We evaluate how TrADe's target contamination factor estimate (and that of the baselines) affects the target detector's anomaly detection performance through the following experiment: (i) pick one of the 206 source-target pairs; (ii) use TrADe or one of the baselines to estimate the target contamination factor; (iii) compute the target anomaly scores using an anomaly detector on the target domain; (iv) use the estimated contamination factor to convert the anomaly scores to hard predictions and compute the F_1 score. To avoid the results being dependent on one specific anomaly detector, we repeat the experiment for each of the 9 considered detectors resulting in $206 \times 9 = 1854$ experiments. We compute the F_1 score because it *strictly* depends on using the target contamination factor γ_m^T to make hard predictions. In contrast, the AUC metrics commonly used in anomaly detection [33], only evaluate a detector's capability to rank examples correctly and do not change when γ_m^T changes.

Table 4.1 (bottom) summarizes the results of the F_1 score obtained using the target contamination factor estimated by TrADe and the baselines in each of the 1854 experiments. TrADe has the lowest (best) average F_1 rank. On average, TrADe enables the anomaly detector to achieve higher/similar F_1 scores in at least 60% of the experiments.

Figure 4.2 (right) shows TrADe's average improvement in F_1 score compared to the baselines aggregated for each of the 23 target domains. Positive values indicate that TrADe obtains higher F_1 scores. TrADe results in higher average F_1 scores on 17 target domains vs. Otsu, 18 vs. Source$_\lambda$, 20 vs. Unify, 21 vs. Source$_\gamma$, and 22 vs. Coral.

Q3. Ensemble versus single anomaly detectors. Our method uses an ensemble of anomaly detectors to estimate the target contamination factor and set the threshold. To see the effect of this choice, we compare TrADe using the ensemble with variants of TrADe using only one of the nine detectors. For computational reasons, this experiment only considers the water and wind turbine data. Compared to using a single detector, the ensemble results in an equivalent or better estimate of the contamination factor on between 59% (vs. IF variant) to 85% (vs. HBOS variant) of the experiments. Overall, the ensemble variant reduces the MAE from 12% (vs. IF variant) to 50% (vs. kNNO variant).

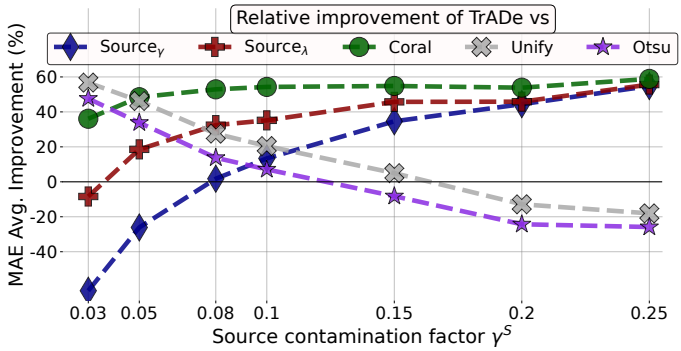

Figure 4.3: TrADe's relative improvement in MAE versus each baseline as a function of the source contamination factor on the IoT datasets. As the gap between the source and target contamination factors increases, TrADe performance gains versus Source$_\gamma$, Source$_\lambda$, and Coral grow.

Q4. The effect of varying the source contamination factor γ^S. In the IoT dataset, the target contamination is always 0.01. Therefore we explore the effect on performance of varying the source contamination factor. Figure 4.3 reports the TrADe's average improvement in MAE over the baselines as a function of the source contamination factor. Because Source$_\gamma$ and Source$_\lambda$ depend on the source γ^S, TrADe achieves better results when γ^S increases. Compared to these methods, TrADe's performance is not as adversely affected by increasing the difference between the source and the target contamination factors. Because Unify and Otsu are unsupervised

methods using only the target domain, their estimate is constant as it does not depend on the source contamination factor. TrADe results in (large) gains over Unify and Otsu even for relatively large gaps between the source and target contamination factor (e.g., 0.01 for the target and 0.10 for the source). As the gap between the source and target contamination factor grows, TrADe win in performance vs. Unify and Otsu shrinks, with the two baselines outperforming TrADe at the largest gaps.

Benefits and limitations. In the experiments we focused on anomaly detection in a sustainability context (preventing water losses in retail stores and blade icing in wind turbines) and security. The potential societal benefits, due to the more accurate detection models, are manifested in the avoidance of potentially costly anomalies (e.g., large water leaks). A potential downside would arise from missed detections and false alarms, which both result in real-world costs. Moreover, one could our approach to disadvantage or discriminate against marginalized groups, indicating them as anomalies.

4.4 Related Work

Although no existing research attempts at transferring the contamination factor between similar domains, two main research lines are connected to our task.

Combining transfer learning and anomaly detection. A first related research line looks at combining transfer learning with anomaly detection in different application domains. For instance, time series anomaly detection [244], detecting dangerous aircraft test flight actions [246], hyperspectral image anomaly detection [143], or video anomaly detection [14, 152]. Some authors focus on instance-transfer for anomaly detection [235, 234], others on feature-based transfer [136, 248], or model-based transfer [240, 111, 55]. The goal is almost always to improve a target model using source domain label information, i.e., deriving better estimates for the anomaly scores. However, none of these works look neither at transferring the contamination factor between domains nor at setting a prediction threshold on the target anomaly scores.

Calibrating the anomaly scores. A second related research line revolves around converting anomaly scores into calibrated probabilities [74]. Although calibration usually requires either labeled examples or a *known* contamination factor, Kriegel et al. Kriegel et al. [129] introduce Unify, a method to obtain calibrated probabilities from anomaly scores without such requirements. In absence of labeled data, Marques et al. [154] develop an internal measure to evaluate the quality of an anomaly detector,

while Schubert et al. [208] and Perini et al. [183] develop rank similarity measures to compare the anomaly rankings of different detectors. However, none of these works propose a method to find an appropriate decision threshold for the anomaly scores in an (unlabeled) dataset.

4.5 Conclusion

In this Chapter, we explored the setting where one monitors multiple different yet related assets (e.g., a fleet of wind turbines) and wants to perform anomaly detection on each of them. Specifically, our goal was to estimate the contamination factor of a target domain when provided with a source dataset with a known contamination factor. Our key insight was that the distribution of anomaly scores for normal examples in both domains will show some similarity when derived from the same anomaly detection algorithm. Our theoretical analysis confirmed that the contamination factor estimated by our method TrADe converges to its true value as the size of the target dataset increases. Through empirical experiments, we illustrated that TrADe outperforms several baselines adapted to estimate the target contamination factor. Moreover, we showed that a more accurate estimate yields a higher F_1 score.

In the next Chapter, we investigate another setting, namely when one is given just an unlabeled dataset. For this goal, we move from a frequentist (i.e., aiming for a point-wise estimate of the contamination factor) to a Bayesian perspective, where the goal becomes estimating the contamination's posterior distribution.

Chapter 5

Estimating the contamination factor's distribution in unsupervised anomaly detection

Estimating the contamination factor γ with no domain knowledge is challenging. One can directly threshold the scores through statistical threshold estimators, and derive γ as the proportion of scores higher than the threshold. For instance, the Modified Thompson Tau test thresholder (MTT) finds the threshold through the modified Thompson Tau test [202], while the Inter-Quartile Region thresholder (IQR) uses the third quartile plus 1.5 times the inter-quartile range [15].[1]

However, all existing methods rely on data-driven intuitions about the score distributions and often the estimators are not accurate. Moreover, transforming the scores into predictions using an incorrect estimate of the contamination factor (or, equivalently, an incorrect threshold) deteriorates the anomaly detector's performance [70, 63] and reduces the trust in the detection system. If such an estimate was coupled with a measure of uncertainty, one could take into account this uncertainty to improve decision making. Although existing methods propose Bayesian anomaly detectors [214, 204, 108, 102] to assign anomaly scores, none of them study how to transform the scores into hard predictions. Therefore, this Chapter investigates the estimation of the contamination factor from a Bayesian perspective.

[1] In Section 5.2 we provide a comprehensive list of threshold estimators.

Problem Statement

The problem that this chapter addresses is the following:

Given: an unlabeled dataset D and a set of M unsupervised anomaly detectors $\{f_1, \ldots, f_M\}$;

Do: Estimate a (posterior) distribution of the contamination factor $p(\gamma|S)$.

Contributions of this Chapter

This chapter proposes γGMM, the first algorithm for estimating the contamination factor's (posterior) distribution in unlabeled anomaly detection setups. First, we use a set of unsupervised anomaly detectors to assign anomaly scores for all samples and use these scores as a new representation of the data. Second, we fit a Bayesian Gaussian Mixture model with a Dirichlet Process prior (DPGMM) [65, 198] in this new space. If we knew which components contained the anomalies, we could derive the contamination factor's posterior distribution as the distribution of the sum of such components' weights. Because we do not know this, as a third step γGMM estimates the probability that the k most extreme components are jointly anomalous, and uses this information to construct the desired posterior. The method is explained in detail in Section 5.1.

In summary, we make four contributions. First, we adopt a Bayesian perspective and introduce the problem of estimating the contamination factor's posterior distribution. Second, we propose an algorithm that is able to sample from this posterior. Third, we demonstrate experimentally that the implied uncertainty-aware predictions are well calibrated and that taking the posterior mean as point estimate of γ outperforms several other algorithms in common benchmarks. Finally, we show that using the posterior mean as a threshold improves the actual anomaly detection accuracy.

The content of this chapter is based on the following publication [181]:[2]

> Perini, L., Bürkner, P.-C., and Klami, A. Estimating the contamination factor's distribution in unsupervised anomaly detection. In *Proceedings of the Fortieth International Conference on Machine Learning (ICML 2023)*, pp 27668–27679, PMLR.

[2] LP provided the main body of the work (code, text), PB assisted in designing the proper Bayesian framework, AK guided formalizing and structuring the text.

5.1 Methodology

Learning from an unlabeled dataset has three key challenges. First, the absence of labels forces us to make relatively strong assumptions. Second, the anomaly detectors rely on different heuristics that may or may not hold, and their performance can vary significantly across datasets. Third, we need to be careful in introducing user-specified hyperparameters, because setting them properly may be as hard as directly specifying the contamination factor.

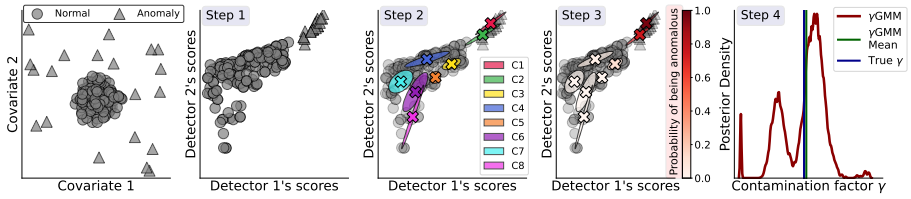

Figure 5.1: Illustration of the γGMM's four steps on a 2D toy dataset (left plot): we 1) map the 2D dataset into an $M = 2$ dimensional anomaly space, 2) fit a DPGMM model on it, 3) compute the components' probability of being anomalous (conditional, in the plot), and 4) derive $\gamma|S$'s posterior. γGMM's mean is an accurate point estimate for the true value γ^*.

We propose γGMM, a novel Bayesian approach that estimates the contamination factor's posterior distribution in four steps, which are illustrated in Figure 5.1:

Step 1. Because anomalies may not follow any particular pattern in covariate space, γGMM maps the examples into an M dimensional anomaly space, where the dimensions correspond to the anomaly scores assigned by the M unsupervised anomaly detectors. Within each dimension of such a space, the evident pattern is that "the higher the more anomalous".

Step 2. We model the examples in the new space $\mathbb{R}^M$ using a Dirichlet Process Gaussian Mixture Model (DPGMM) [168, 198]. We assume that each of the (potentially many) mixture components contains either only normals or only anomalies. If we knew which components contained anomalies, we could then easily derive γ's posterior as the sum of the mixing proportions π of the anomalous components. However, such information is not available in our setting.

Step 3. Thus, we order the components in decreasing order of anomalousness, and we estimate the probability of the largest k components being anomalous. This poses three challenges: (a) how to represent each M-dimensional component by a single value to sort them from the most to the least anomalous, (b) how to compute the probability that the kth component is anomalous given that the $(k-1)$th is, (c) how to derive the target probability that k components are jointly anomalous.

Step 4. γGMM estimates the contamination factor's posterior by exploiting such a joint probability and the components' mixing proportions posterior.

In the following, we describe these steps in detail.

5.1.1 Representing Data Using Anomaly Scores

Learning from an unlabeled anomaly detection dataset has two major challenges. First, anomalies are rare and sparse events, which makes it hard to use common unsupervised methods like clustering [29]. Second, making assumptions on the unlabeled data is challenging due to the absence of specific patterns in the anomalies, which makes it hard to choose a specific anomaly detector.

Therefore, we use a set of M anomaly detectors to map the d-dimensional input space into an M-dimensional score space $\mathbb{R}^M$, such that a sample x gets a score s:

$$\mathbb{R}^d \ni x \rightarrow [f_1(x), f_2(x), \ldots, f_M(x)] = s \in \mathbb{R}^M.$$

This has two main effects: (1) it introduces an interpretable space where the evident pattern is that, within each dimension, higher scores are more likely to be anomalous, and (2) it accounts for multiple inductive biases by using multiple arbitrary anomaly detectors.

To make the dimensions comparable, we (independently for each dimension) map the scores s to $\log(s - \min(S) + 0.01)$, where $\min(S)$ is estimated taking the minimum score over the training examples. Note that the log is used to shorten heavy right tails. Finally, we normalize them to have zero mean and unit variance.

5.1.2 Modeling the Density with DPGMM

We use mixture models as basis for quantifying the distribution of the contamination factor, relying on their ability to model the proportions of samples using the mixture weights. For flexible modeling, we use the DPGMM

$$\begin{aligned} s_i &\sim \mathcal{N}(\tilde{\mu}_i, \tilde{\Sigma}_i) \qquad i = 1, \ldots, N \\ (\tilde{\mu}_i, \tilde{\Sigma}_i) &\sim G \\ G &\sim DP(G_0, \omega) \\ G_0 &= \mathcal{NIW}(M, \lambda, V, u) \end{aligned}$$

where G is a random distribution of the mean vectors μ_i and covariance matrices Σ_i, drawn from a DP with base distribution G_0. We use the explicit representation $G =$

$\sum_{k=1}^{\infty} \pi_k \delta_{(\mu_k,\Sigma_k)}(\tilde{\mu}_i, \tilde{\Sigma}_i)$, where $\delta_{(\mu_k,\Sigma_k)}$ is the delta distribution at (μ_k, Σ_k) and π_k follow the stick-breaking distribution. We set G_0 as Normal Inverse Wishart [174] with parameters M, λ, V, u common to all components. We use variational inference (VI; see e.g. Blei et al. [26] for details) for approximating the posterior as VI is computationally efficient and sufficiently accurate for our purposes. Alternative methods (e.g., Markov Chain Monte Carlo [30]) could also be used but were not considered worth the additional computational effort here.

Choice of DPGMM. DPGMM has two key properties that justify its use over other flexible density models [87, 147]. First, we choose Gaussian distributions over more robust heavy-tailed distributions because isolated samples are likely candidates for outliers, and encouraging the model to represent them using the heavy tails would be counter-productive. Second, the rich-get-richer property of DPs is desirable because we expect some very large components of normals but want to allow arbitrarily small clusters of anomalies. Moreover, the DP formulation allows us to refrain from specifying the number of components K. After fitting the model, we only consider the components with at least one observation assigned to them and propagate all the remaining density uniformly over the active components. Thus, for the following steps we can still proceed as if the model was a finite mixture with π following a Dirichlet distribution.

5.1.3 Estimating the Components' Anomalousness

We assume that each mixture component either contains only anomalous or only normal samples. All unsupervised methods rely on some assumption on nearby samples sharing latent characteristics, and this cluster assumption is a natural and weak assumption. If we knew which components contain anomalies, we could directly derive the posterior of the contamination factor γ as the sum of the mixing proportions π_k of those components. This is naturally not the case, but we need to estimate it in an unsupervised fashion.

More formally, we estimate the probability that k (out K) components are anomalous such that we can derive γ's posterior by averaging over all the values $0 \leq k \leq K$. We do this in three steps. Initially, we sort the components of score vectors in decreasing order (by degree of anomalousness), which comes natural from the representation we made in Step 1 (Sec. 5.1.1). Then, our insight is that the kth component can be anomalous only if the $(k-1)$th is such. This points to the estimation of conditional probabilities, i.e., the probability of c_k = "**the kth component is anomalous**" given c_{k-1}. Finally, the probability that exactly the first k components are anomalous can be obtained using basic rules of probability theory.

Assigning an ordering to the components. As initial step for computing the joint probability, we need to design a decreasing ordering map for the components based on their anomalousness. We do this in a manner that accounts for the uncertainty of the components' parameters to rank high the components that can be reliably identified as anomalous: we want the means to be high but the variance low, to avoid the risk that also samples with low anomaly scores could belong to the component.

We construct the overall ranking using dimension-specific scores because our normalization cannot remove all statistical differences between the different detectors. Formally, let $r\colon \mathbb{R}^M \times \mathbb{R}^{M\times M} \to \mathbb{R}$ be the function of the mean vector μ_k and the covariance matrix Σ_k that assigns a real value representing the component k's anomalousness. We set r as

$$r\left(\mu_k^{(z)}, \Sigma_k^{(z)}\right) = \frac{1}{M}\sum_{j=1}^{M} \frac{\mu_k^{j(z)}}{1 + \sqrt{\Sigma_k^{j,j(z)}}}, \tag{5.1}$$

where $\mu_k^{(z)}$ and $\Sigma_k^{(z)}$ are samples from the parameters' posterior distributions of the kth component. We obtain a representative value of the whole component by taking the expected value of r, i.e. through $\mathbb{E}[r(\mu_k, \Sigma_k)]$. Equation (5.1) intentionally does not consider inter-dimension correlations, as it remains unclear to us how those should ideally be included and what benefits it would actually provide.

We add 1 to the component's standard deviation for two reasons. First, if a component contains samples with almost the same covariate values, the standard deviation would be close to 0 and the ratio would explode towards infinity, masking any effect of the mean. Second, adding 1 is reasonable because it is equal to the theoretical upper bound of the components' variances, as they are normalized (Sec. 5.1.1).

Without loss of generality, from now on we assume that the components' index k is ordered based on their representative value such that the kth component has a higher value (i.e., more anomalous) than the $(k+1)$th component.

Estimating the probability that the kth component is anomalous. Because the components are sorted by anomalousness, our key insight is that *the kth component can be anomalous only if the* $(k-1)$*th is anomalous*. Formally,

$$\mathbb{P}(c_k|\, c_{k-1}) > 0 \;\;\&\;\; \mathbb{P}(c_k|\, \bar{c}_{k-1}) = 0 \quad (1 < k \leq K)$$

where $\bar{c}_{k-1}$ means "not c_{k-1}". Moreover, we assume $\mathbb{P}(c_1) \in (0, 1)$. That is, we allow for the data to not have anomalies (< 1) but exclude certain knowledge of no anomalies (> 0). This is a sensible assumption because, if one knew for sure that no anomalies are in the data, then we trivially have $\gamma = 0$, whereas we still need to allow for the data to be free of anomalies if evidence suggests so.

We estimate the conditional probability as

$$\mathbb{P}(c_k|c_{k-1}) = \frac{1}{1 + e^{(\psi+\delta\cdot r(\mu_k,\Sigma_k))}}, \tag{5.2}$$

where ψ and δ are the two hyperparameters of the sigmoid function, which will be discussed in Section 5.1.4. Note that the principle itself is not restricted to this particular choice of functional form. One could apply any transformation that maps to [0, 1], but the detailed derivations of the parameters would naturally be different.

Deriving the components' joint probability. Given the conditional probability $\mathbb{P}(c_k|\, c_{k-1})$, the joint probability follows from simple steps. Taking inspiration from the sequential ordinal models [32], our insight is that exactly k components are jointly anomalous if and only if each of them is conditionally anomalous and the $(k + 1)$th is not anomalous. We indicate this as $C^* = k$. Essentially,

$$\mathbb{P}(C^* = k) \coloneqq \mathbb{P}(c_1, \ldots, c_k, \bar{c}_{k+1}, \ldots, \bar{c}_K) = \mathbb{P}(c_1) \prod_{t=1}^{k-1} \mathbb{P}(c_{t+1}|c_t)(1 - \mathbb{P}(c_{k+1}|c_k)) \tag{5.3}$$

for any $k \leq K$, where $\mathbb{P}(c_{K+1}\,|c_K) = 0$ by convention.

5.1.4 Estimating the Contamination Factor's Distribution

Given the joint probability that the first k components are anomalous (for $k \leq K$), the contamination factor γ's posterior distribution can be obtained as

$$p(\gamma|S) = \sum_{k=1}^{K} p(C^* = k) \cdot p\left(\sum_{j=1}^{k} \pi_j \Big| S\right) \tag{5.4}$$

where $p(\sum_{j=1}^{k} \pi_j|S)$ is the posterior distribution of the sum of the first k components' mixing proportions, $p(C^* = k)$ are densities WRT the counting measure. Note that $p(\sum_{j=1}^{k} \pi_j|S) = \textsc{Beta}(\sum_{j=1}^{k} \omega_j, \sum_{j=k+1}^{K} \omega_j)$, if $p(\pi_1, \ldots, \pi_K|S) = \textsc{Dir}(\omega_1, \ldots, \omega_K)$ [147].

Setting the sigmoid's hyperparameters ψ and δ. Introducing new hyperparameters when the task is to estimate the contamination factor γ's posterior is risky because setting their value may be as difficult as directly providing a point estimate of γ. Our key insight is that we can obtain ψ and δ by asking the user two simple questions: (a) How likely is that no anomalies are in the data? (b) How likely is it that a large number of anomalies occurred, say, more than $t = 15\%$ of the data? Both of these values are

supposed to be low. Let's call p_0 and p_{high} the two answers. Formally,

$$p_0 = 1 - \mathbb{P}(c_1) = 1 - \frac{1}{1 + e^{(\psi + \delta \cdot r(\tilde{\mu}_1, \tilde{\Sigma}_1))}} \tag{5.5}$$

$$p_{\text{high}} = \mathbb{P}(\gamma \geq t|S) = \sum_{k=1}^{K} \mathbb{P}(C^* = k) \cdot \mathbb{P}\left(\sum_{j=1}^{k} \pi_j \geq t|S \right) \tag{5.6}$$

One can use a numerical solver for non-linear equations with linear constraints (e.g., the least square optimizer implemented in SkLearn) to find the values of ψ and δ that satisfy such constraints. The problem has a unique solution whenever $p_{\text{high}} \geq \mathbb{P}(\pi_1 \geq t|S)$. This holds almost always in our experiments, but, in case such a constraint cannot be satisfied, we keep running again the variational inference method (with different starting points) for the DPGMM until the constraint on p_{high} holds. If this cannot happen or does not happen within 100 iterations, we reject the possibility of too high contamination factors and just set it to 0. In the experiments (Q5), we show that changing the p_0 and p_{high} does not have a large impact on γ's posterior.

Sampling from γ's posterior. Our estimate of the contamination factor's posterior $p(\gamma|S)$ does not have a simple closed form. However, we can sample from the distribution using a simple process. The DPGMM inference determines an approximation for $p(\pi, \mu, \Sigma|S)$ and all the quantities required for Equations (5.2), (5.3), (5.4) can be computed based on samples from the approximation. Formally, we derive a sample from $p(\gamma|S)$ in four steps by repeating the next operations for all $k \leq K$. First, we draw a sample $\pi_k^{(z)}, \mu_k^{(z)}, \Sigma_k^{(z)}$ from π_k (Dirichlet), μ_k (Normal), Σ_k (Inverse Wishart). Second, we transform $\pi_k^{(z)}$ by taking the cumulative sum and obtain a sample $\sum_{j=1}^{k} \pi_j^{(z)}$. Third, we pass $\mu_k^{(z)}$ and $\Sigma_k^{(z)}$ through the sigmoid function (5.2) to get the conditional probabilities $\mathbb{P}(c_k \mid c_{k-1})$, and transform them into the exact joint probabilities $\mathbb{P}(C^* = k)$ using the equation 5.3. Finally, we multiply the samples following Formula 5.4 and obtain a sample $\gamma^{(z)}$ from $p(\gamma|S)$. We show the pseudo-code of our approach in Algorithm 4.

Additional technical details. Because our method uses the variational inference approximation, we run it 10 times and concatenate the samples to reduce the risk of biased distributions due to local minima. Moreover, after sorting the components, we set $\mathbb{P}(c_k|c_{k-1}) = 0$ for all $k > K' = \arg\max\{k\colon\ \mathbb{E}[\sum_{j=1}^{k} \pi_j] < 0.25\}$. This has the effect of setting an upper bound of 0.25 to the contamination factor γ. Because anomalies must be rare, we realistically assume that it is not possible to have more than 25% of them. Although "0.25" could be considered a hyperparameter, this value has virtually no impact on the experimental results. Moreover, note that $\mathbb{E}[\pi_1] \geq 0.25$ cannot occur, as otherwise we could not set the hyperparameters p_0 and p_{high}.

A final pseudo-code summarizing our approach γGMM is shown in Algorithm 4.

Algorithm 4 Pseudo-code for the algorithm of γGMM.

Input: A dataset D of size N; M unsupervised detectors $f_1, \ldots, f_M$; the prior parameters λ, V, u, ω; the hyperparameters p_0 and p_{high}.
Output: $\gamma^{(z)}$, a sample from $p(\gamma|S)$.

$s \leftarrow [f_1(x), f_2(x), \ldots, f_M(x)]$ for each $x \in D$
for m in $[1, \ldots, M]$ **do**
    $s_m \leftarrow \log(s_m - \min\{s_m\} + 0.01)$ // preprocess scores
end for
K, $p(\pi, \mu, \Sigma|S) \leftarrow$ FIT_DPGMM($\{s\}, \lambda, V, u, \omega$)
$\bar{\mu}, \bar{\Sigma} \leftarrow$ COMPUTE_PARAMETERS_MEAN($p(\mu, \Sigma|S), K$)
for k in $[1, \ldots, K]$ **do**
    $r_k \leftarrow$ COMPUTE_REPRESENTATIVE_VAL($\bar{\mu}_k, \bar{\Sigma}_k$) // see Eq. 5.1
end for
$k_1, \ldots, k_K \leftarrow$ ARGSORT($r_1, \ldots, r_K$, ORDER = "descending")
$\psi, \delta \leftarrow$ OPTIMIZE(p_0, p_{high}) // see Eq. 5.5, 5.6
$\pi^{(z)}, \mu^{(z)}, \Sigma^{(z)} \leftarrow$ SAMPLE($p(\pi, \mu, \Sigma|S)$, SIZE = 1)
for k_i in $[k_1, \ldots, k_K]$ **do**
    $\pi^{(z)}_{k_i} \leftarrow \sum_{j=1}^{k_i} \pi^{(z)}_{k_j}$ // compute cumulative
    $r^{(z)}_{k_i} \leftarrow$ COMPUTE_REPRESENTATIVE_VAL($\mu^{(z)}_{k_i}, \Sigma^{(z)}_{k_i}$)
    $\mathbb{P}(c_{k_i}|c_{k_i-1}) \leftarrow$ COMPUTE_CONDITIONAL_PROB($\psi, \delta, r^{(z)}_{k_i}$) // see Eq. 5.2
    $\mathbb{P}(C^* = k_i) \leftarrow$ COMPUTE_JOINT_PROB($\mathbb{P}(c_{k_i}|c_{k_i-1})$) // see Eq. 5.3
end for
$\gamma^{(z)} \leftarrow$ COMPUTE_GAMMA_SAMPLE($\{\mathbb{P}(C^* = k)\}, \{\pi^{(z)}_k\}$) // see Eq. 5.4
return $\gamma^{(z)}$

5.2 Experiments

We empirically evaluate two aspects of our method: (a) whether it accurately estimates the contamination factor's posterior, and (b) how thresholding the scores using our method affects the anomaly detectors' performance. To this end, we address the following five experimental questions:

Q1. Is the posterior estimate sharp and well-calibrated?

Q2. How does γGMM compare to threshold estimators?

Q3. Does a better point estimate of γ improve the anomaly detector performance?

Q4. What is the impact of the number of detectors M?

Q5. How sensitive the method is to p_0 and p_{high}?

5.2.1 Experimental Setup

Methods. We compare the sample mean of γGMM[3] with 21 threshold estimators that we cluster into 9 groups:
1. Kernel-based. FGD [193] and AUCP [201] both use the kernel density estimator to estimate the score density; FGD exploits the inflection points of the density's first derivative, while AUCP uses the percentage of the total kernel density estimator's AUROC to set the threshold;
2. Curve-based. EB [72] creates elliptical boundaries by generating pseudo-random eccentricities, while WIND [114] is based on the topological winding number with respect to the origin;
3. Normality-based. ZSCORE [12] exploits the Z-scores, DSN [9] measures the distance shift from a normal distribution, and CHAU [27] follows the Chauvenet's criterion before using the Z-score;
4. Regression-based. CLF and REGR [4] are two regression models that separate the anomalies based on the y-intercept value;
5. Filter-based. FILTER [98], and HIST [226] use the wiener filter and the Otsu's method to filter out the anomalous scores;
6. Statistical test-based. GESD [8], MCST [43] and MTT [202] are based on, respectively, the generalized extreme studentized, the Shapiro-Wilk, and the modified Thompson Tau statistical tests;
7. Statistical moment-based. BOOT [160] derives the confidence interval through the two-sided bias-corrected and accelerated bootstrap; KARCH [3] and MAD [11] are based on means and standard deviations, i.e., the Karcher mean plus one standard deviation, and the mean plus the median absolute deviation over the standard deviation;
8. Quantile-based. IQR [15] and QMCD [113] set the threshold based on quantiles, i.e., respectively, the third quartile Q_3 plus 1.5 times the inter-quartile region $|Q_3 - Q_1|$, and the quantile of one minus the Quasi-Monte Carlo discreprancy;
9. Transformation-based. MOLL [122] smooths the scores through the Friedrichs' mollifier, while YJ [200] applies the Yeo-Johnson monotonic transformations.

We apply each threshold estimator to the univariate anomaly scores of each detector at a time. *We average the contamination factors over the M detectors and use it as the final point estimate for each dataset.*

Data. We carry out our study on 20 commonly used benchmark datasets and additionally 2 (proprietary) real tasks. The benchmark datasets contain semantically useful anomalies widely used in the literature [33]. The datasets vary in size, number of features, and true contamination factor. The Supplement B provides further details. For the real tasks, our experiments focus on preventing blade icing in wind turbines.

[3]Code is available at: `https://github.com/Lorenzo-Perini/GammaGMM`

We use two public wind turbine datasets, where sensors collect various measurements (e.g., wind speed, power energy, etc.) every 7 seconds for either 8 weeks (turbine 15) or 4 weeks (turbine 21). Following [254], we construct feature-vectors by taking the average over the time segment of one minute.

Evaluation metrics. We use three evaluation metrics to assess the performance of the methods. Contrary to all the threshold estimators, our method estimates the posterior of γ. Therefore, we measure the **probabilistic calibration** of γGMM's posterior using a QQ-plot with the x-axis representing the expected probabilities and on the y-axis the empirical frequencies (called reliability diagram). That is, for $v \in [0, 0.5]$,

$$\text{Expected Prob.} = \mathbb{P}\left(\gamma^* \in [q(0.5 - v), q(0.5 + v)]\right) = 2v$$

$$\text{Empirical Freq.} = \frac{|\{\gamma \in [q(0.5 - v), q(0.5 + v)]\}|}{\#\text{experiments}},$$

where $q(u)$ is the quantile at the value u of our distribution, for $u \in [0, 1]$, and γ^* refers to the true dataset's contamination factor. For evaluating the point estimate of the methods, we use the **mean absolute error** (MAE) between the method's point estimate and the true value. Finally, we measure the impact of thresholding the scores using the methods' point estimate through the F_1 score [88], as common AUC metrics are not affected by different thresholds. Specifically, for $m = 1, \ldots, M$, we measure the **relative deterioration of the F_1 score**:

$$F_1 \text{ deterioration} = \frac{F_1(f_m, D, \gamma^*) - F_1(f_m, D, \hat{\gamma})}{F_1(f_m, D, \hat{\gamma})}$$

where we compute the F_1 score on the dataset D using the anomaly detector f_m, and either the true value γ^* or an estimate $\hat{\gamma}$ to threshold the scores. The F_1 deterioration of a method is (mostly) negative, and the higher the better.

Setup. In the experiments we assume a transductive setting [33, 209, 227], where a dataset D is used both for training and testing. This is the typical setting of anomaly detection [29, 207, 10, 151] because the absence of labels and patterns (for the anomaly class) avoids overfitting issues.

For each dataset, we proceed as follows: (i) use a set of M anomaly detectors to assign the anomaly scores S to each observation in the dataset D; (ii) map each anomaly score $s \in S$ to $\log(s - \min(S) + 0.01)$ and normalize them to have mean equal to 0 and standard deviation equal to 1; (iii) either use our method to estimate the contamination factor's posterior and extract the posterior mean as point estimate $\hat{\gamma}$, or use one of the threshold estimators to directly obtain a point estimate $\hat{\gamma}$ of the contamination factor (see methods paragraph above); (iv) evaluate the point estimates using the

mean absolute error (MAE) between such estimate and the true value γ^*; (v) use the contamination factor's point estimate to threshold the anomaly scores of each of the M anomaly detectors f_m (individually); (vi) finally, we measure the F_1 score and compute the relative deterioration.

Hyperparameters, anomaly detectors and priors. Our method introduces two new hyperparameters: p_0 and p_{high}. We set both of them to 0.01 as default value because extremely high contamination, as well as no anomalies, are unlikely events. We will experimentally check the impact of these two hyperparameters in Q5.

We use 10 anomaly detectors with different inductive biases (see Sec. 2.2.3 for details) [220]: kNNO [10], IF [151], LOF [29], OCSVM [89], AE [39], VAE [123], LSCP [257], HBOS [82], LODA [189], and COPOD [145]. All these methods are implemented in the Python library PyOD [258].

The threshold estimators are implemented in PYTHRESH[4] with default hyperparameters. Finally, the DPGMM is implemented in SKLEARN: we use the Stick-breaking representation [60], with 100 as the upper bound of K. We set the means' prior to 0, and the covariance matrices' prior to identities of appropriate dimension. We opt for such (in our context) weakly informative priors because sensible prior knowledge of the DPGMM hyperparameters is hard to come by in practice.

5.2.2 Experimental Results

Q1. Does our method estimate a sharp and well-calibrated posterior of γ? Figure 5.2 shows the contamination factor γ's posterior estimated by our method on the 22 datasets. In several cases (e.g., WPBC, Cardio, SpamBase, Wilt, and T21), the distribution looks accurate as γ's true value (blue line) is close to the posterior mean (i.e., the expected value, the green line). On the contrary, some datasets (e.g., Arrhythmia, Shuttle, KDDCup99, Parkinson, Glass) obtain less accurate distributions: although γ's true value sometimes falls on low-density regions (Arrhythmia, Shuttle), in many cases it would be quite likely to sample the true value from our posterior (KDDCup99, Parkinson, Glass), which makes the density still quite reliable.

Figure 5.3 shows the calibration plot. The posterior is well-calibrated as it is very close to the dashed black line indicating a perfectly calibrated distribution. The empirical frequencies deviate from the real probabilities by less than 5% (dark shadow grey) in more than 76% of the cases, while never deviating by more than 10% (light shadow grey).

[4]Link: `https://github.com/KulikDM/pythresh`.

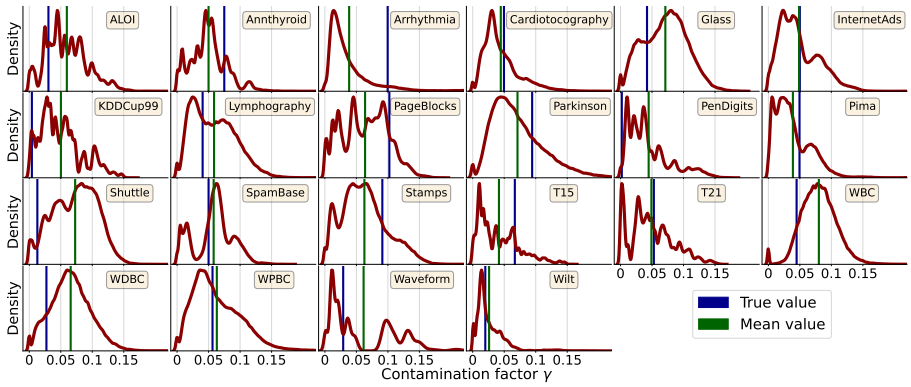

Figure 5.2: Illustration of how γGMM estimates γ's posterior distribution (red) on the 22 datasets. The blue vertical line indicates the true contamination factor, while the green line is the posterior's mean.

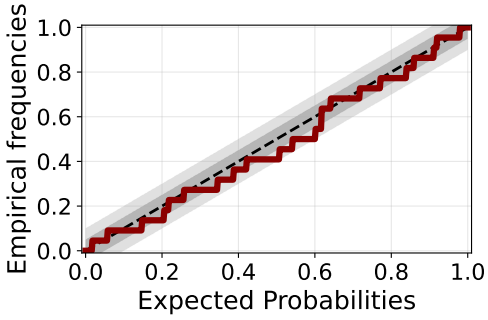

Figure 5.3: QQ-plot of γGMM's distribution estimate. The black dashed line illustrates the perfect calibration, while shades indicate a deviation of 5% (dark) and 10% (light) from the black line.

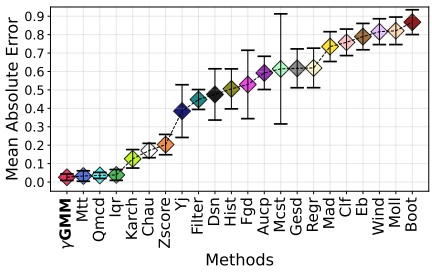

Figure 5.4: Average MAE (± std.) of γGMM's sample mean compared to the other methods. Our method has the lowest (better) average, which is 20% lower than the runner-up.

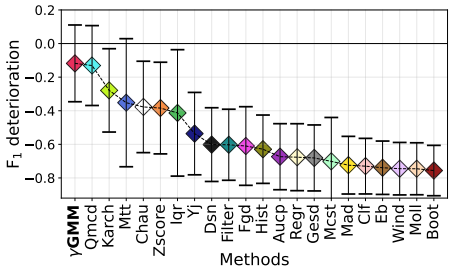

Figure 5.5: F_1 deterioration (mean ± std) for each method, where the higher the better. γGMM ranks as best method, obtaining $\approx$ 10% higher average than the runner-up QMCD.

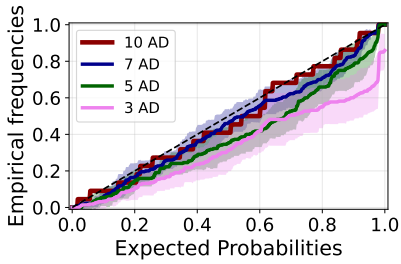

Figure 5.6: QQ-plot comparing the calibration curves of γGMM when a different number M of detectors is used. Colored shades report the uncertainty obtained by randomly sampling the detectors. The higher the number of detectors, the more calibrated the distribution.

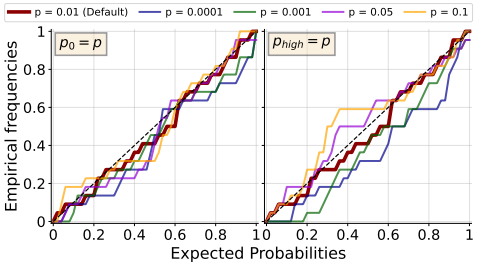

Figure 5.7: QQ-plot showing how calibrated γGMM's posterior mean would be if we varied p_0 (left) and p_{high} (right). While p_0 does not have a large impact on the method, the empirical frequencies slightly under (over) estimate the expected probabilities for low (high) values of p_{high}.

Q2. How does γGMM compare to the threshold estimators? We take γGMM's posterior mean as our best point estimate of γ and compare such value to the point estimates obtained from the threshold estimators. Figure 5.4 illustrates the ordered MAE (mean ± std.) between the methods' estimate and the true γ. On average, γGMM obtains a MAE of 0.026 that is 20% lower than the best runner-up MTT and 27% lower than the third best method QMCD (MAE of 0.033 and 0.036). For each experiment, we rank the methods from the best (position 1, lowest MAE) to the worst (position 22, greatest MAE). Our method has the best average rank (2.13 ± 1.04). Moreover, γGMM ranks first 8 times ($\approx$ 36% of the cases), and for 13 times ($\approx$ 60% of the cases) it is in the top two. The next best method, MTT, ranks first in 6 cases with an average rank of 2.30 ± 1.10.

Q3. Does a better contamination improve the anomaly detectors' performance? We use γGMM's posterior mean as a point estimate to measure the F_1 score of the anomaly detectors because sampling from the distribution would not imply a fair comparison against the other methods that can only provide a point estimate. Moreover, anomaly detectors that fail to rank the samples accurately perform poorly even when using the correct γ. Since our focus is studying the effect of γ, for each dataset D, we compare F_1 scores only over the detectors that achieve the greatest F_1 score using the true contamination factor γ^*, i.e. $\arg\max_{f_m} \{F_1(f_m, D, \gamma^*)\}$. The Supplement B contains the list of detectors used for each experiment.

Figure 5.5 shows the average (± std.) deterioration for each of the methods. On average, γGMM has the best F_1 deterioration (-0.117 ± 0.228) that is around 10% better than the runner-up QMCD (-0.131 ± 0.238), and 58% better than the next best KARCH (-0.279 ± 0.248). For 25% of the cases we get a higher F_1 score with γGMM than when using the true γ^*. This is due to the (still incorrect) ranks made by the detectors, which achieve better performance with slightly incorrect contamination factors. The Supplement B provides further details on how the methods perform in terms of false alarms and false negatives.

Q4. What is the impact of M on γ's posterior? In the previous experiments, we used $M = 10$ detectors. We evaluate the effect of M by running all the experiments 10 times with (different) randomly chosen detectors for $M = 3, 5, 7$. Figure 5.6 shows that the calibration suffers if using fewer detectors, but already $M = 5$ let the method work fairly well. The variance of the results (over repeated experiments) also increases for lower M.

Q5. Impact of the hyperparameters p_0 and p_{high}. We evaluate the impact of p_0 and p_{high} by running the experiments with smaller and larger values than 0.01: we vary,

one at a time, p_0, $p_{\text{high}} \in [0.0001, 0.001, 0.05, 0.1]$ and keep the other set as default. Figure 5.7 shows the QQ-plot for p_0 (left) and p_{high} (right). In both cases, smaller hyperparameters lead to slightly under-estimated expected probabilities. Overall, our method is robust to different values of p_0, while p_{high} affects the calibration slightly more. Comparing the resulting 8 variants of γGMM in terms of MAE, we conclude that the posterior means produce similar values to our default setting, obtaining an MAE that varies from 0.252 ($p_{\text{high}} = 0.001$, the best) to 0.32 ($p_0 = 0.0001$, the worst).

5.3 Conclusion

In this Chapter, we investigated how to estimate the contamination factor from a Bayesian perspective, namely how to derive its posterior distribution. For this task, we introduced γGMM, which works in four steps: (i) it creates an M dimensional score-based representation of the training examples by using M anomaly detectors, (ii) it fits a DPGMM on such representation, (iii) it computes the probability that a component of the DPGMM is anomalous, and (iv) it derives the posterior of the contamination factor by leveraging such probabilities and the DPGMM's weights distributions. By running experiments on 22 benchmark and real-world datasets, we showed that (1) γGMM's posterior distribution is well calibrated, and (2) using γGMM's posterior mean as a point-wise estimate effectively yields more accurate contamination factors than most baselines. Moreover, we proved that better estimates of γ yield better thresholds being picked, which, in turn, improve the performance of an anomaly detector (F_1 score).

Setting the threshold using the contamination factor introduces uncertainty in the model's predictions. That is, because the value of the threshold strictly relates to the training anomaly scores, slight changes in the training set yield different thresholds being set. As a consequence, some (test) predictions $\hat{y}$ may flip, even for the same example x. In the next Chapter, we investigate this phenomenon and propose to quantify such uncertainty, which we call *stability*.

Chapter 6

Quantifying an anomaly detector's example-wise stability

Being the anomalies rare and unpredictable, learning a reliable unsupervised anomaly detector is challenging. Hypothetically, even if we could collect multiple datasets, each one would contain distinct anomalies to which the anomaly detectors would assign different scores. Additionally, small perturbations in the training data might cause (large) differences in an example's anomaly score and, consequently, a different prediction. Consider the three one-dimensional toy datasets in Figure 6.1. The middle row plots show the continuous anomaly scores that κNNO and IF assign to each example in the distributions. These scores change as a result of small perturbations in the dataset, ultimately resulting in different predictions.

Problem Statement

The problem that this chapter addresses is the following:

Given: An unlabeled dataset D with N examples, the contamination factor $\gamma \in [0, 1]$ and an anomaly detector f;

Do: Design a stability metric $\mathscr{S}$ that captures how the example-wise predictions of f change under slight perturbations of D.

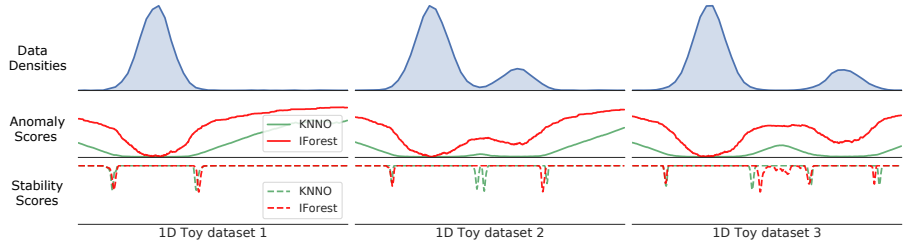

Figure 6.1: Illustration of why interpretable scores are important on three 1D toy datasets. The top plots show the data distributions under small perturbations. The middle plots show the anomaly scores assigned by kNNO and IF. These two models produce non-standard scores, which are difficult to interpret and compare. The bottom plots show the corresponding stability scores computed using our method. Small changes in the data distribution affect anomaly scores and predictions. The stability scores capture clearly where the models (dis)agree. The dips in the stability scores correspond to a transition in the predicted label of the underlying model.

Contributions of this Chapter

This Chapter tackles this challenge by providing a measure of how *stable* an anomaly detector's predictions are on an example-wise basis. The measure will allow a user to assess the reliability of anomaly detectors in different scenarios.

In summary, we make the following four contributions. First, we propose a notion of a stability measure that captures how consistent a model's prediction would be for that example if the training data were perturbed. Second, we propose ExCeeD (*EXample-wise ConfidEncE of anomaly Detectors*), an approach that is able to compute our stability metric for any anomaly detector that produces a real-valued anomaly score. The method begins by transforming the anomaly scores to *outlier probabilities* using a Bayesian approach. Then, it uses these probabilities to derive the example-wise stability scores. Third, we perform a theoretical analysis of the convergence behaviour of our confidence estimates. Fourth, we perform an extensive empirical evaluation on 21 benchmark datasets.

The content of this chapter is based on the following publication [187]:[1]

> Perini, L., Vercruyssen, V., and Davis, J. Quantifying the confidence of anomaly detectors in their example-wise predictions. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2020)*, pp 227–243, Springer.

[1] LP provided the main body of the work (code, theory, text), VV assisted with experiments and nailing down the right research questions, JD guided formalizing and structuring the text.

6.1 Methodology

Using an anomaly detector in practice requires converting its returned anomaly score into a hard prediction. Typically, this is done by setting a threshold λ on the scores. Then, any example x with a score $s > \lambda$ will be classified by the model as an anomaly. Hence, perturbing the training data would lead to a different threshold being picked, which in turn would affect an example's predicted class.

To capture this potential uncertainty, we propose a notion of a detector's example-wise stability in its predictions, which works with any anomaly detector producing real-valued scores. Intuitively, we can think of example-wise stability as the probability that a detector's prediction would change if a different dataset was observed. More formally, we define it as follows.

Definition 5 (Example-wise Stability)**.** *Let f be the anomaly score function that maps examples to anomaly scores. Given an unlabeled dataset D, the model's stability for an example x with anomaly score $s = f(x)$ is defined as*

$$\mathscr{S}(\hat{Y})_x = \begin{cases} \mathbb{P}(\hat{Y}^* = 1 \mid s, N, \gamma, \hat{p}_s) & \text{if } \hat{Y} = 1 \\ 1 - \mathbb{P}(\hat{Y}^* = 1 \mid s, N, \gamma, \hat{p}_s) & \text{if } \hat{Y} = 0 \end{cases} \tag{6.1}$$

where $\mathbb{P}$ is the probability computed over the possible datasets D^* with N examples, $\hat{Y}^*$ refers to the prediction label when the model is trained on D^*, $\hat{p}_s$ is the estimated outlier probability (i.e., the probability that the example belongs to the anomaly class), and γ is the expected proportion of positive examples.

From now on, when we use the term *stability* we will refer to $\mathbb{P}(\hat{Y}^* = 1 \mid s, N, \gamma, \hat{p}_s)$, as the case when $\hat{Y} = 0$ is directly computable from the previous one. Hence, when $\hat{Y} = 1$, high values of $\mathbb{P}(\hat{Y}^* = 1 \mid s, N, \gamma, \hat{p}_s)$ indicate that model is stable in its prediction that the example is an anomaly. One would expect stability values around 0.5 when an example is near the decision boundary, that is on the border between normal and abnormal behaviors. We estimate the stability in two steps. First, we employ a Bayesian approach to estimate the distribution of anomaly scores. This allows us to derive an example's outlier probability $\hat{p}_s$. Second, we use the outlier probability to estimate the stability of the anomaly detector by considering how the combination of the observed training set and contamination factor γ would be used to select the threshold λ for converting anomaly scores into predictions.

Definition 6 (Outlier Probability)**.** *Given an anomaly detector f, for any example $x \in \mathbb{R}^d$ we define the* outlier probability *of x as the probability that x is anomalous according to its anomaly score $s = f(x)$*

$$\mathbb{P}(Y = 1 | f(X) = f(x)) = \mathbb{P}(Y = 1 | S = s) \coloneqq \mathbb{P}(S \leq s). \tag{6.2}$$

Subsequently, the probability that one example is normal can be computed as

$$\mathbb{P}(Y = 0|f(X) = f(x)) = 1 - \mathbb{P}(Y = 1|S = s) = \mathbb{P}(S > s).$$

6.1.1 A Bayesian Approach for Assigning Outlier Probability

Our goal is to infer the true class label based on the anomaly score. Formally, for any score $s \in \mathbb{R}$, we can model the example's true class given the score as the conditional random variable $Y|S = s$. Based on this framework, we can estimate an example's *outlier probability* (i.e., the probability it belongs to the anomaly class) as follows:

$$P_s \coloneqq \mathbb{P}(Y = 1|S = s) = \mathbb{P}(S \leq s).$$

Because $Y|S = s$ takes values in the set $\{0, 1\}$, we model its outcome using a Bernoulli distribution:

$$Y|S = s \sim Bernoulli(P_s)$$

where P_s is the probability of success. If we knew S's distribution, we could compute $\mathbb{P}(S \leq s)$ – the probability that an example belongs to the anomaly class –using the cumulative distribution function of S. Unfortunately, the distribution of S is usually unknown which makes it infeasible to directly approximate P_s.

Our solution is to take a Bayesian approach to this problem. The key insight is to measure the area of $\{S < s\}$ by drawing samples from the real distribution. We will view P_s as a random variable and assume a uniform prior. Theoretically, we can derive the probability that an example belongs to the anomaly class as follows. First, we draw one example a from the distribution of S, which simply entails drawing an example x from X and computing its anomaly score $a = f(x)$. Second, we record the event as a success (i.e., $b = 1$) if $a \leq s$ and as failure (i.e., $b = 0$) otherwise. We repeat the process N times and record the total number of successes as t and failures as $N - t$. In fact, the rate between successes and trials, corrected with other factors, will approximate the outlier probability as defined in formula 6.2. Thanks to Bayes's rule we can use the following theorem [66].

Theorem 6 (Fink [66])**.** *Assume that a random variable P_s follows a Beta distribution $\mathcal{B}eta(\alpha, \beta)$ as prior. Given the events $b_1, \ldots, b_N$, which are i.i.d. examples drawn from a Bernoulli random variable $Bernoulli(P_s)$, then the posterior distribution of P_s is still a Beta distribution with new parameters $\mathcal{B}eta\,(\alpha + t, \beta + N - t)$, where $t = \sum_{i=1}^{N} b_i$ is the number of successes.*

Proof. According to the hypotheses, the prior distribution of P_s has density

$$p(q) = \frac{q^{\alpha-1}(1-q)^{\beta-1}}{\mathcal{B}(\alpha, \beta)},$$

where $\mathcal{B}(\alpha, \beta)$ is the Euler beta function. So, by using the Bayes's rule

$$p(q|b_1, \ldots, b_t) = \frac{p(q) \cdot p(b_1, \ldots, b_t|q)}{\int_0^1 p(r) \cdot p(b_1, \ldots, b_t|r)\, dr} = \frac{\frac{q^{\alpha-1}(1-q)^{\beta-1}}{\mathcal{B}(\alpha,\beta)} \cdot q^t\,(1-q)^{N-t}}{\int_0^1 \frac{r^{\alpha-1}(1-r)^{\beta-1}}{\mathcal{B}(\alpha,\beta)} \cdot r^t\,(1-r)^{N-t}\, dr}$$

$$= \frac{q^{\alpha+t-1}\,(1-q)^{\beta+N-t-1}}{\int_0^1 r^{\alpha+t-1}\,(1-r)^{\beta+N-t-1}\, dr} \implies P_s|b_1, \ldots, b_t \sim \mathcal{B}eta\,(\alpha + t, \beta + N - t)$$

where $t = \sum_{i=1}^{N} b_i$, $p(q|b_1, \ldots, b_N)$ is the posterior distribution of P_s after i.i.d. sampling N *Bernoulli*(P_s) examples, and $p(b_1, \ldots, b_N|q)$ is the likelihood. □

In our setting we assume a uniform prior, i.e. that $P_s \sim \mathcal{B}eta(1, 1) = Unif(0, 1)$. As a result, the posterior distribution of P_s is still a Beta distribution

$$P_s|b_1, \ldots, b_N \sim \mathcal{B}eta\,(1 + t, 1 + N - t)\,. \tag{6.3}$$

In order to derive an estimate of the outlier probability from P_s, we take the expectation of P_s. Since the posterior distribution is known from (6.3), its expectation can be obtained as a function of the parameters:

$$\hat{p}_s \coloneqq \mathbb{E}[P_s|b_1, \ldots, b_N] = \frac{1 + t}{2 + N}. \tag{6.4}$$

In practice, we cannot sample from the true distribution and instead need to use the training scores $\{s_1, \ldots, s_N\}$ to infer the posterior distribution. Thus, when drawing an example, we are restricted to sampling from the dataset. This limits us to drawing N examples, that is, the total number of examples in the dataset. An additional consideration concerns the value of t. It represents the number of successes when sampling from $Y|S = s \sim Bernoulli(P_s)$. As a result, t is a practical approximation of the real percentage θ of successes times the number of trials N

$$t \approx \theta \cdot N. \tag{6.5}$$

The reason why we use t is to obtain a corrected estimate of the real parameter θ, which would be the exact probability value of $Y|S = s$ if it were known.

6.1.2 Deriving a Detector's Stability in its Predictions

Although the second step of our framework works with any approach that converts anomaly scores into outlier probabilities, here $\hat{p}_s$ refers to the definition in Equation 6.4. Deriving the stability value requires estimating the proportion of times that an example

will be predicted as being anomalous by the chosen anomaly detection algorithm. This requires analyzing how to set the threshold λ for converting anomaly scores to predictions. Typically, anomaly detectors exploit the contamination factor γ to pick the threshold λ. Note that γ may be known from domain knowledge (e.g., historical anomaly rates) or it can be estimated from partially labeled data (see Section 3). There are two different scenarios:

$\gamma \in (0, 1)$**: The training set contains some anomalies.** Here, the standard approach is to compute the expected number of anomalies in the training set as $k = \gamma \times N$.[2] Then, it ranks the training examples by their anomaly scores and sets the threshold to be the value in position k.

$\gamma = 0$**: The training set contains only normal examples.** In this case, the threshold has to be equal to the maximum anomaly score in the training set. Picking a lower value would result in a false positive on the training data.

In both cases, the chosen threshold depends on the distribution of anomaly scores in the training set. In turn, these scores depend on the available data sample. That is, if we drew another training set from the population, the chosen threshold may change. This leads to our key insight: *the task of measuring the model's stability can be formulated as estimating the probability that an example with score s will be classified as an anomaly based on a theoretical sample $\{s_1, \ldots, s_N\}$ drawn from the population*. Formally, given the training set size N and the contamination factor γ, we want to compute the probability that an example x with score $s = f(x)$ and outlier probability $\hat{p}_s$ will be classified as an anomaly when randomly drawing a training set of N examples from the population of scores. In practice, the stability can be seen as the probability that the chosen threshold λ value will be less than or equal to the score s. This probability depends on our two cases for picking the threshold:

Contamination factor $\gamma \in (0, 1)$. In this case by drawing theoretically from the Bernoulli distribution of $Y|S = s$ with parameter P_s, we should get at least $N - k + 1$ successes to classify s as anomaly, where $k = \gamma \times N$ and "success" means that the drawn value is lower than s. As a result, our stability is defined as:

$$\mathbb{P}(\hat{Y}^* = 1 \mid s, N, \gamma, \hat{p}_s) = \sum_{i=N(1-\gamma)+1}^{N} \binom{N}{i} \hat{p}_s^i (1 - \hat{p}_s)^{N-i} \tag{6.6}$$

where $\hat{p}_s = \mathbb{E}[P_s | b_1, \ldots, b_N]$, which is estimated using our Bayesian approach for computing an example's outlier probability (see Equation 6.4). Hence, our stability estimate explicitly relies on our outlier probability.

[2] We assume that $k \in \mathbb{N}$, taking the floor function when needed.

Contamination factor $\gamma = 0$. In this case, the threshold is the maximum score in the training set, $\lambda = \max\{s_i\}_{i=1}^{N}$. We need to compute the probability that an example with score s and outlier probability $\hat{p}_s$ will be classified as an anomaly when randomly drawing N examples from the normal population. It is quite similar to the previous case, with the only difference being that no failures[3] are allowed. So, when $\gamma = 0$ we need to denote the stability as

$$\mathbb{P}(\hat{Y}^* = 1 \mid s, N, 0, \hat{p}_s) = (\hat{p}_s)^N \tag{6.7}$$

where again $\hat{p}_s = \mathbb{E}[P_s|b_1, \ldots, b_N]$ comes from our Bayesian estimate of the example's outlier probability (see Equation 6.4.)

The pseudo-code in Algorithm 5 shows our method in a summary.

Algorithm 5 Pseudo-code for the algorithm of ExCeeD.

Input: A dataset D of size N with contamination factor γ; an unsupervised detector f; a test example x^*.
Output: $\mathscr{S}_{x^*}$, the detector's stability on x^*.

```
s ← f(D)                                          // training anomaly scores
λ ← SET_DECISION_THRESHOLD({s}, 1 − γ)
s* ← f(x*)                                        // test anomaly score
t = |{s ≤ s*}|
p̂_s ← COMPUTE_OUTLIER_PROB(t, N)                  // see Eq. 6.4
if γ > 0 then
    S_x* = COMPUTE_STABILITY(p̂_s, γ, N)           // see Eq. 6.6
else
    S_x* = COMPUTE_STABILITY(p̂_s, N)              // see Eq. 6.7
end if
if s* < λ then
    S_x* = 1 − S_x*
end if
return S_x*
```

6.2 Convergence Analysis of our Stability Estimate

This section analyzes the behavior of our stability estimate. In particular, given a fixed anomaly score s for a test example, we want to investigate how our stability in the model's prediction changes as the number of training examples tends towards infinity.

[3] A failure would correspond to a training example having a higher anomaly score than the chosen threshold. Given the assumption that all training examples are normal, this would indicate a false positive.

We would expect that as the size of the training set increases, our stability estimate should converge. Again, we analyze the two cases based on whether or not the training set contains any anomalies.

6.2.1 Convergence Analysis when $\gamma \in (0, 1)$

In this case, when we set the threshold based on a fixed set of training scores $\{s_1, \ldots, s_N\}$, we can derive our stability for a test example with score s by merging Eq. 6.4 and Eq. 6.6:

$$\mathbb{P}(\hat{Y}^* = 1|s, N, \gamma, \hat{p}_s) = \sum_{i=N(1-\gamma)+1}^{N} \binom{N}{i} \left(\frac{1+t}{2+N}\right)^i \left(\frac{1+N-t}{2+N}\right)^{N-i} \tag{6.8}$$

where t represents the successes in the Bayesian learning phase (section 6.1.1).

This leads to the question: how does the stability about the class prediction for a score s behave as the number of training examples N goes towards $+\infty$? In order to formally analyze this, we rewrite Eq. 6.8 as

$$\mathbb{P}(\hat{Y}^* = 1|s, N, \gamma, \hat{p}_s) = F_T(N) - F_T(N - \gamma N)$$

where the sum in Eq. 6.8 is the cumulative distribution of a binomial random variable $T \sim \mathcal{B}(N, \gamma N, \hat{p}_s)$ with N trials, γN successes, and probability $p = \hat{p}_s$. When N increases, the central limit theorem yields [67]:

$$\mathbb{P}\left(\frac{T - N\hat{p}_s}{\sqrt{N\hat{p}_s(1-\hat{p}_s)}} \leq c\right) \to \Phi(c) \qquad \text{for } N \to +\infty,\ \forall c \in \mathbb{R}.$$

Consequently, assuming that N is large enough, we assert that, $\forall c \in \mathbb{R}$ [221],

$$\mathbb{P}\left(T \leq c\right) = F_T(c) \approx \Phi\left(\frac{c - N\hat{p}_s + 0.5}{\sqrt{N\hat{p}_s(1-\hat{p}_s)}}\right),$$

where +0.5 is a correction due to the continuity of the Gaussian variable. Thus, the stability can be approximated by the cumulative distribution function of a Gaussian variable T^* with mean $\mu = N\hat{p}_s + 0.5$ and variance $\sigma^2 = N\hat{p}_s(1 - \hat{p}_s)$,

$$T^* \sim \mathcal{N}(N\hat{p}_s + 0.5, N\hat{p}_s(1 - \hat{p}_s)).$$

Therefore:

$$\mathbb{P}(\hat{Y}^* = 1|s, N, \gamma, \hat{p}_s) = F_T(N) - F_T(N - \gamma N)$$

$$\approx \Phi\left(\frac{N - N\hat{p}_s + 0.5}{\sqrt{N\hat{p}_s(1-\hat{p}_s)}}\right) - \Phi\left(\frac{N(1-\gamma) - N\hat{p}_s + 0.5}{\sqrt{N\hat{p}_s(1-\hat{p}_s)}}\right)$$

$$= \mathbb{P}\left(N(1-\gamma) \leq T^* \leq N\right) = \mathbb{P}\left((1-\gamma) \leq \frac{T^*}{N} \leq 1\right).$$

Next, we analyze the behaviour of $\frac{T^*}{N}$ as $N \to \infty$ in order to interpret the final result. Since $N \in \mathbb{N}$, it still follows a normal distribution with new parameters:

$$\mathbb{E}\left[\frac{T^*}{N}\right] = \frac{1}{N}\mathbb{E}[T^*] = \hat{p}_s - \frac{1}{2N} = \frac{1+t}{2+N} - \frac{1}{2N};$$

$$\mathrm{Var}\left[\frac{T^*}{N}\right] = \frac{1}{N^2}\mathrm{Var}[T^*] = \frac{\hat{p}_s(1-\hat{p}_s)}{N} = \frac{1+N-t}{N(2+N)}.$$

Since the mean and the variance of $\frac{T^*}{N}$ are bounded (they both are decreasing sequences when N increases), the sequence of Gaussian random variables $\frac{T^*}{N}$ converges in distribution to a Gaussian random variable parameterized by the limit of the mean and the limit of the variance, respectively:

$$\lim_{N\to\infty} \mathbb{E}\left[\frac{T^*}{N}\right] = \theta, \quad \lim_{N\to\infty} \mathrm{Var}\left[\frac{T^*}{N}\right] = 0,$$

where θ is defined in Eq. 6.5. Hence, when $N \to +\infty$, the limit random variable is normally distributed with mean θ and variance 0, which means the only value it assumes is θ, the true outlier probability (see the end of Section 6.1.1). Formally, calling the limit degenerate random variable θ^*,

$$\lim_{N\to\infty} \mathbb{P}(\hat{Y}^* = 1|s, N, \gamma, \hat{p}_s) = \mathbb{P}\left((1-\gamma) \leq \theta^* \leq 1\right).$$

Intuitively, this means that as the number of training examples goes to infinity the population is perfectly estimated and represented by the sample, which yields two cases. In the first case, the true outlier probability θ is greater than $1-\gamma$. Roughly speaking, the expected proportion of normal examples $(1-\gamma)$ is not high enough to yield a threshold value less than the considered score s. Hence, the stability will be 1, because $\mathbb{P}\left((1-\gamma) \leq \theta^* \leq 1\right) = 1$ since θ^* takes the constant value θ and the inequalities are satisfied. In contrast, in the second case the inequalities are not respected, meaning that θ does not fall inside the interval $[1-\gamma, 1]$. Hence, in this scenario the proportion of normal examples is such that the value of the threshold must be greater than s and, as a result, the stability is 0 when predicting class 1.

At the end, the limit of the stability that s is predicted to be an anomaly is

$$\lim_{N\to\infty} \mathbb{P}(\hat{Y}^* = 1|s, N, \gamma, \hat{p}_s) = \begin{cases} 1 & \text{if } \theta \geq 1-\gamma; \\ 0 & \text{if } \theta < 1-\gamma. \end{cases}$$

This corresponds to our intuition of what should occur when given an infinite number of training examples.

6.2.2 Convergence Analysis when $\gamma = 0$

In this analysis, the main hypothesis is that the training set only contains normal examples, which corresponds to learning a one-class model. Hence, only a representative sample of the normal class can be used to train the model.

The problem we tackle is: Given a true anomaly with score s^*, how stable will the model be in predicting that it belongs to the anomaly class? Our intuitions might be misleading in this case. In fact, since the contamination factor is 0, the threshold will be set as the highest observed score in the training set and the definition of stability slightly changes. In this case no failures are allowed (i.e., all training examples must have a score less than the chosen threshold), once one training example has a score greater than s^*, this implies that the chosen threshold will be greater than s^* as well. In practice, if s^* is always greater than or equal to the anomaly score for each training example, the model will always predict that the test example is anomalous but its stability might not be so high. The reason is simple: since the sample of training scores $\{s_1, \ldots, s_N\}$ represents the normal class, the model cannot learn from the anomalies. So, drawing a normal example with a high anomaly score is theoretically possible. Hence, for a fixed anomalous test example, we need to analyze how the model's stability in its prediction for the example changes as the number of normal examples in the training increases.

Theorem 7. *Given a set of training scores $\{s_1, \ldots, s_N\}$ and a score s^* such that $s_i < s^*$ $\forall i \leq N$, fixed $\gamma = 0$, the expected rate of anomalies in $\{s_1, \ldots, s_N\}$, then*

$$\mathbb{P}(\hat{Y}^* = 1|s^*, N, \gamma = 0, \hat{p}_{s^*}) \longrightarrow \frac{1}{e} \approx 0.368 \quad \textit{for } N \to +\infty.$$

Proof. Assuming that $\{s_1, \ldots, s_N\}$ contains no anomalies, we get $t = N$ successes. This yields an outlier probability of:

$$\hat{p}_{s^*} = \mathbb{E}[P_{s^*}|b_1, \ldots, b_t] = \frac{1+t}{2+N} = \frac{1+N}{2+N}.$$

Then, using the estimated probability that one score is less than or equal to s^*, we can compute the stability using the hypothesis that the contamination factor in the training

set is 0, meaning that no failures are allowed:

$$\mathbb{P}(\hat{Y}^* = 1 | s^*, N, \gamma = 0, \hat{p}_{s^*}) = (\hat{p}_{s^*})^N .$$

In fact, if we drew a score greater than s^* from the training set, then the threshold would be greater than s^* (predicted class equal to 0). Let's now analyze the limit:

$$\lim_{N \to +\infty} \mathbb{P}(\hat{Y}^* = 1 | s^*, N, \gamma = 0, \hat{p}_{s^*}) = \lim_{N \to +\infty} (\hat{p}_{s^*})^N = \lim_{N \to +\infty} \left(\frac{1+N}{2+N} \right)^N$$

$$= \lim_{N \to +\infty} \left(\frac{2+N-1}{2+N} \right)^N = \lim_{N \to +\infty} \left[\left(1 + \frac{-1}{2+N} \right)^{2+N} \left(\frac{1+N}{2+N} \right)^{-2} \right] = \frac{1}{e},$$

where the first factor is a notable limit and converges to $\frac{1}{e}$, whereas the second term converges to 1 because of the rate of polynomials of degree 1. □

This can be understood as follows. While the outlier probability for a true anomaly goes to 1 when $N \to +\infty$, the number of normal examples in the training data also increases. Thus, we are more likely to observe unlikely events, i.e., the training set containing a normal example with a high anomaly score.

6.3 Experiments

The goal of our empirical evaluation is to: (1) intuitively illustrate how our stability score works; (2) evaluate the quality of our stability scores; and (3) assess the effect of using our Bayesian approach for converting anomaly scores to outlier probabilities on the quality of the stability scores.

6.3.1 Experimental Setup

Our experimental goal is to evaluate ExCeeD's ability to recover the example-wise stability of an anomaly detector as opposed to evaluating predictive performance or quality of calibration [180]. Hence, metrics like the AUROC or Brier score are not suitable for assessing how small perturbations in the training data affect the example-wise predictions of the detector. For instance, AUROC would treat models that flipped the positions of two anomalies (normals) in the ranking produced by each model as being equivalently performant. In contrast, we are explicitly interested in understanding the number and magnitude of such flips. Moreover, our approach for converting the anomaly score to an outlier probability is a monotone function and hence does not affect the AUROC of the model. We consider a stability score to be good if it accurately

Dataset	N	d	γ	Dataset	N	d	γ
ALOI	12384	27	0.030	PenDigits	9868	16	0.002
Annthyroid	7129	21	0.075	Pima	625	8	0.200
Arrhythmia	450	259	0.457	Shuttle	1013	9	0.013
Cardiotocography	1734	21	0.050	Spambase	3160	57	0.200
Glass	214	7	0.042	Stamps	340	9	0.091
HeartDisease	270	13	0.444	Waveform	3443	21	0.029
Hepatitis	80	19	0.163	WBC	454	9	0.022
Ionosphere	351	32	0.359	WDBC	367	30	0.027
Lymphography	148	19	0.040	Wilt	4655	5	0.020
PageBlocks	5473	10	0.102	WPBC	198	33	0.237
Parkinson	60	22	0.200				

Table 6.1: The 21 benchmark anomaly detection datasets from [33] and their characteristics: number of examples (N), number of features (d), and contamination γ.

captures the consistency with which a detector predicts the same label for an example. Hence, we expect a detector to predict for all examples subject to a stability value of $\mathscr{S}(\hat{Y})_x$ the same label $\mathscr{S}(\hat{Y})_x$-percent of the time when retraining the detector multiple times with slightly perturbed training datasets. Therefore, we propose a novel method for evaluating an anomaly detector's example-wise stability as an indication of how consistently it predicts the same label for that example. The method (1) draws 1000 sub-samples from the training data with the size of each sub-sample randomly selected in $[0.2 \cdot N, N]$, (2) trains an anomaly detector on each sub-sample, (3) uses each detector to predict the class labels of every example in the test set, and finally (4) computes for each test example x the frequency F_x with which the detector predicted the same class.

We carry out our study on a benchmark consisting of 21 standard anomaly detection datasets from [33]. The datasets vary in size, number of features, and proportion of anomalies (Table 6.1). Given a benchmark dataset, we can now evaluate our stability scores as follows. First, we split the dataset into training and test sets with stratified 5-fold cross-validation. Then, we take the class-weighted average of the L^2 differences between the stability score $\mathscr{S}(\hat{Y})_x$ and the earlier computed frequency F_x of each test set example x, yielding:

$$error(\mathscr{S}, F) = \frac{1}{2|T_{\mathcal{N}}|} \sum_{x \in T_{\mathcal{N}}} \left(\mathscr{S}(\hat{Y})_x - F_x\right)^2 + \frac{1}{2|T_{\mathcal{A}}|} \sum_{x \in T_{\mathcal{A}}} \left(\mathscr{S}(\hat{Y})_x - F_x\right)^2$$

where $T_{\mathcal{N}}$ are the true test set normals and $T_{\mathcal{A}}$ the true test set anomalies. We take a class-weighted average because the large class-imbalances that characterize anomaly detection datasets, would otherwise skew the final error. We report averages over the folds. The underlying anomaly detector is either κNNO, IF, or OCSVM. This results in $21 \times 3 = 63$ experiments for each method. We compare 9 approaches, which can be divided into three categories:

Our method. **ExCeeD** as introduced in Section 6.1.[4]

Naive baselines. **ExCeeD-m**, this is ExCeeD with a different prior distribution $\mathcal{B}eta(\gamma \cdot m, m \cdot (1-\gamma))$ where γ is the contamination factor and m is such that $\gamma \cdot m = 10$ (suggested in [250]). A **Baseline** approach which assumes the stability scores to be equal to the model's predictions.

Outlier probability methods. The Unify [129], linear, and squash methods for estimating the outlier probabilities $\hat{p}_s$ from anomaly scores. These probabilities are *not* stability scores. To obtain true stability scores, we have to combine each of these methods with the second step of our framework, yielding **ExCeeD-Unify**, **ExCeeD-Linear**, and **ExCeeD-Squash**.

Calibration methods. The **Logistic** [190], **Isotonic** [251], and **Beta** [135] methods to empirically estimate calibration frequencies. Although these methods do not compute stability scores, probabilistic predictions are often (incorrectly) interpreted as such and therefore included for completeness.

Method	$error(\mathscr{S}, F)$ **rank**	**# times ExCeeD**			$error(\mathscr{S}, F)$
	Mean ± SD	**Wins**	**Loses**	**Draws**	**Mean ± SD** $\times 10^2$
ExCeeD	**1.429 ± 0.844**	-	-	-	**1.972 ± 2.637**
Baseline	2.270 ± 0.641	**52**	3	8	2.679 ± 2.931
ExCeeD-Unify	4.103 ± 2.040	**55**	2	6	15.13 ± 16.29
Isotonic	5.151 ± 1.482	**59**	2	2	17.92 ± 13.68
Beta	5.421 ± 1.285	**62**	0	1	23.13 ± 29.65
Logistic	5.532 ± 1.525	**62**	0	1	23.89 ± 29.69
ExCeeD-m	5.937 ± 2.709	**59**	1	3	24.95 ± 37.38
ExCeeD-Linear	6.802 ± 1.350	**61**	2	0	31.47 ± 23.15
ExCeeD-Squash	8.357 ± 1.271	**61**	2	0	45.97 ± 36.67

Table 6.2: Comparison of ExCeeD with the baselines. The table shows: the weighted average $error(\mathscr{S}, F)$ rank ± standard deviation (SD) of each method; the weighted average $error(\mathscr{S}, F)$ ± SD of each method (computed as in [51]); and the number of times ExCeeD wins (lower error), draws, and loses (higher error) against each baseline.

6.3.2 Experimental Results

To illustrate the intuition behind our approach, Figure 6.2 compares how the stability scores computed by the different methods evolve as we gradually move an example from the large normal central cluster to the small anomaly cluster in the bottom right (using kNNO). For ExCeeD, the stability scores start out high when the example is

[4] Implementation available in PyOD: `https://pyod.readthedocs.io.html`.

close to the cluster of normal points and the prediction is 0 (i.e., normal). The stability gradually decreases as the example moves away from the normal cluster, eventually reaching about 50% when it is halfway between the normal and abnormal clusters. Once the example is far enough away from the normal cluster, the stability increases again as the model changes its prediction and becomes more certain that the example is anomalous. Using ExCeeD with its Bayesian outlier probability clearly captures the gradual change in stability we would intuitively expect in this scenario.

Question 1: Does ExCeeD produce good stability scores? Table 6.2 summarizes the comparison between ExCeeD and the baselines in terms of $error(\mathscr{S}, F)$. Our method outperforms all baselines and has the lowest average error rank over the 63 experiments. It also achieves lower errors in at least 54 of the 63 experiments compared to every other method and achieves the lowest weighted average error. When the results are split out per underlying anomaly detector (Table 6.3), ExCeeD still outperforms all baselines, winning against each baseline at least 20, 18 and 13 out of 21 times when the detectors are, respectively, kNNO, IF and OCSVM.

Question 2: Does the Bayesian approach to estimate outlier probabilities contribute to better stability scores? Our stability score can be computed from any outlier probability measure. To evaluate specifically how our proposed Bayesian approach for estimating the outlier probabilities contributes to the stability scores, we simply compare ExCeeD with the ExCeeD-Unify, ExCeeD-Linear, and ExCeeD-Squash baselines. The results are summarized in Tables 6.2 and 6.3. The Bayesian subroutine of ExCeeD to compute the outlier probabilities outperforms the three baselines by a substantial margin, obtaining lower errors in respectively 55, 61, and 61 out of 63 experiments, indicating its effectiveness.

Method	kNNO: # of			IF: # of			OCSVM: # of		
	W	D	L	W	D	L	W	D	L
ExCeeD	**-**	-	-	-	-	-	-	-	-
Baseline	**21**	0	0	**18**	0	3	**13**	3	5
ExCeeD-Unify	**21**	0	0	**18**	0	3	**16**	2	3
Isotonic	**20**	0	1	**21**	0	0	**18**	2	1
Beta	**21**	0	0	**21**	0	0	**20**	0	1
Logistic	**21**	0	0	**21**	0	0	**20**	0	1
ExCeeD-m	**20**	0	1	**19**	0	2	**20**	1	0
ExCeeD-Linear	**21**	0	0	**21**	0	0	**19**	2	0
ExCeeD-Squash	**21**	0	0	**21**	0	0	**19**	2	0

Table 6.3: Comparison of ExCeeD with the baselines, split out per anomaly detector (kNNO, IF, and OCSVM). The table presents the number of times ExCeeD wins (W) i.e. lower error, draws (D), and loses (L) i.e. higher error vs. each baseline.

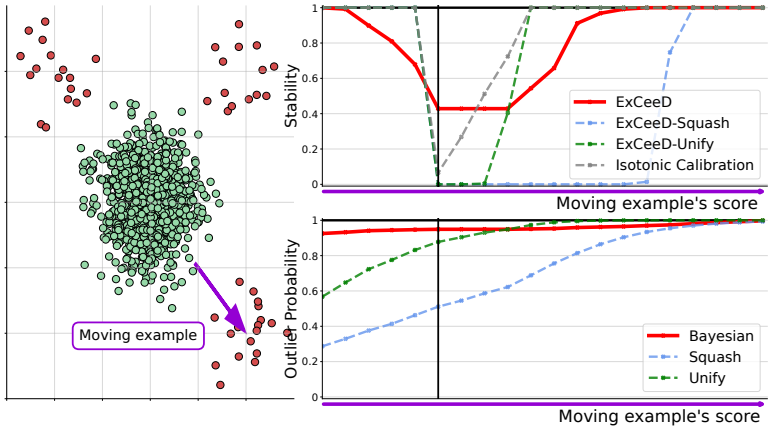

Figure 6.2: Illustration of how moving an example along the purple arrow in the dataset (left plot) affects its outlier probability (bottom-right plot) and its stability score derived from this probability (top-right plot). The underlying anomaly detector is kNNO. Left of the vertical black line, the detector predicts that the example belongs to the normal class. Because at first the example is embedded in the cluster of normal examples (the green points in the dataset), the initial stability score is high. However, the detector's stability in its prediction decreases as the example moves away from the normal points. Finally, it increases again when the example nears the anomalies (the red points) and the example is predicted to be an anomaly. ExCeeD's stability score captures our intuitions that the prediction should be stable (i.e., stability score near 1.0) when the example is very obviously either normal or anomalous and uncertain (i.e., stability score is near 0.5) when the example is equidistant from the normal and anomalous examples.

6.4 Conclusion

We investigated how simulating to draw different training sets (i.e., perturbing the data) affects the model's prediction for a fixed test example. We called our uncertainty measure stability and provided a formal definition, i.e. how likely a model's prediction would flip when subject to perturbations in the training set. Moreover, we introduced ExCeeD, a two-step method to estimate such a stability metric. In the first step, we employed a Bayesian approach to get smooth outlier probabilities. In the second step, these estimated outlier probabilities were utilized to compute a stability score for each example. Through an extensive experimental comparison, ExCeeD showed to recover stability scores that align closer with empirical frequencies than most adapted baselines.

While measuring the model's uncertainty is a hard and important task, exploiting such uncertainty to improve decision-making is even more crucial. That is, how can we leverage the stability metric to enhance an unsupervised anomaly detector? We investigate this question in the following Chapter.

Chapter 7

Unsupervised anomaly detection with rejection

When using an anomaly detector for decision-making, the user must trust the system. However, unsupervised anomaly detectors tend to have high uncertainty, especially close to the decision boundary, because they cannot leverage the labeled examples to distinguish between the two classes. As a result, the detector's predictions should be treated with some circumspection.

One way to increase user trust in the anomaly detection system is to consider Learning to Reject [49]. In this setting, the model does not always make a prediction. Instead, it can abstain when it is at a heightened risk of making a mistake thereby improving its performance when it does offer a prediction. Abstention has the drawback that no prediction is made, which means that a person must intervene to make a decision.

Current approaches for ambiguity rejection threshold what constitutes being too close to the decision boundary by evaluating the model's predictive performance on the examples for which it makes a prediction (i.e., accepted), and those where it abstains from making a prediction (i.e., rejected) [40, 156, 249]. Intuitively, the idea is to find a threshold where the model's predictive performance is (1) significantly lower on rejected examples than on accepted examples and (2) higher on accepted examples than on all examples (i.e., if it always makes a prediction). Unfortunately, existing learning to reject approaches that set a threshold in this manner require labeled data, which is not available in anomaly detection. In this Chapter, we fill in this gap.

Problem Statement

The problem that this chapter addresses is the following:

Given: An unlabeled dataset D with contamination γ, an unsupervised detector f, a cost function $c\colon \{0,1\} \times \{0,1,®\} \to \mathbb{R}$;

Do: Introduce a reject option to f, i.e. find a pair (confidence, threshold) that minimizes the cost.

Contributions of this Chapter

This Chapter proposes an approach to perform ambiguity rejection for anomaly detection in a completely unsupervised manner. Specifically, we make three major contributions. First, we conduct a thorough novel theoretical analysis of a stability metric for anomaly detection [187] and show that it has several previously unknown properties that are of great importance in the context of learning to reject. Namely, it captures the uncertainty close to the detector's decision boundary, and only a limited number of examples get a stability value strictly lower than 1. Second, this enables us to design an ambiguity rejection mechanism without *any labeled data* that offers strong guarantees which are often sought in Learning to Reject [49, 213, 37]. We can derive an accurate estimate of the rejected examples proportion, as well as a theoretical upper bound that is satisfied with high probability. Moreover, given a cost function for different types of errors, we provide an estimated upper bound on the expected cost at the prediction time. Third, we evaluate our approach on an extensive set of unsupervised detectors and benchmark datasets and conclude that (1) it performs better than several adapted baselines based on other unsupervised metrics, and (2) our theoretical results hold in practice.

The content of this chapter is based on the following publication [182]:[1]

Perini, L., and Davis, J. Unsupervised Anomaly Detection with Rejection. In *Proceedings of the Thirty-Seven Conference on Neural Information Processing Systems (NeurIPS 2023).*

[1] LP provided the main body of the work (code, theory, text), JD guided formalizing and structuring the text.

7.1 Methodology

We propose an anomaly detector-agnostic approach for performing learning to reject that requires *no labels*. Our key contribution is a theoretical analysis of the ExCeeD stability-based confidence metric that proves that *only a limited number of examples have confidence lower than* $1 - \varepsilon$ (Sec. 7.1.1). Intuitively, the detector's predictions for most examples would not be affected by slight perturbations of the training set: it is easy to identify the majority of normal examples and anomalies because they will strongly adhere to the data-driven heuristics that unsupervised anomaly detectors use. For example, using the data density as a measure of anomalousness [29] tends to identify all densely clustered normals and isolated anomalies, which constitute the majority of all examples. In contrast, only relatively few cases would be ambiguous and hence receive low confidence (e.g., small clusters of anomalies and normals at the edges of dense clusters).

Our approach is called **RejEx** (Rejecting via ExCeeD) and works in two steps. First, it computes the stability-based confidence metric $\mathscr{M}_s$ as the margin between the two classes' stabilities:

$$\mathscr{M}_s = |\mathbb{P}(\hat{Y}^* = 1|s) - \mathbb{P}(\hat{Y}^* = 0|s)| = |2\mathbb{P}(\hat{Y}^* = 1|s) - 1|.$$

Second, it simply rejects any example with confidence $\mathscr{M}_s$ that falls below threshold $\tau = 1 - \varepsilon$. Theoretically, this constant reject threshold provides several relevant guarantees. First, one often needs to control the proportion of rejections (namely, the *rejection rate*) to estimate the number of decisions left to the user. Thus, we propose *an estimator that only uses training instances to estimate the rejection rate at test time*. Second, because in some applications avoiding the risk of rejecting all the examples is a strict constraint, we provided *an upper bound for the rejection rate* (Sec. 7.1.2). Finally, we compute a *theoretical upper bound for a given cost function* that guarantees that using RejEx keeps the expected cost per example at test time low (Sec. 7.1.3).

7.1.1 Setting the Rejection Threshold through a Novel Theoretical Analysis of ExCeeD

Our novel theoretical analysis proves (1) that the stability metric by ExCeeD is lower than $1 - \varepsilon$ for a limited number of examples (Theorem 8), and (2) that such examples with low confidence are the ones close to the decision boundary (Corollary 9). Thus, we propose to reject all these uncertain examples by setting a rejection threshold

$$\tau = 1 - \varepsilon = 1 - 2e^{-T} \quad \text{for } T \geq 4,$$

where $2e^{-T}$ is the tolerance that excludes unlikely scenarios, and $T \geq 4$ is required for Theorem 8.

We motivate our approach as follows. Given an example x with score s, let $\psi_N = \frac{|\{i \leq N:\ s_i \leq s\}|}{N} \in [0,1]$ be the training frequency, i.e. the proportion of training scores lower than s. Then, Theorem 8 shows that the confidence $\mathscr{M}_s$ is lower than $1 - 2e^{-T}$ (for $T \geq 4$) if ψ_n belongs to an interval $[t_1, t_2]$ obtained as a function of N, γ, T. By analyzing $[t_1, t_2]$, Corollary 9 proves that the closer an example is to the decision boundary, the lower the confidence $\mathscr{M}_s$, and that a score $s = \lambda$ (decision threshold) has confidence $\mathscr{M}_s = 0$.

Remark. Chapter 6 performed an asymptotic analysis of ExCeeD that investigates the metric's behavior when the training set's size $N \to +\infty$. In contrast, this novel analysis is finite-sample and hence provides more practical insights, as real-world scenarios involve having a finite dataset with size $N \in \mathbb{N}$.

Theorem 8 (Analysis of ExCeeD)**.** *Let s be an anomaly score, and $\psi_N \in [0,1]$ its training frequency. For $T \geq 4$, there exist $t_1 = t_1(N, \gamma, T) \in [0,1]$, $t_2 = t_2(N, \gamma, T) \in [0,1]$ such that*

$$\psi_N \in [t_1, t_2] \implies \mathscr{M}_s \leq 1 - 2e^{-T}.$$

Proof. See Appendix C for the formal proof. □

The interval $[t_1, t_2]$ has two relevant groups of properties. First, it becomes *narrower when increasing N* (P1) and *larger when increasing T* (P2). This means that collecting more training data results in smaller rejection regions while decreasing the tolerance $\varepsilon = 2e^{-T}$ has the opposite effect. Second, it is centered (not symmetrically) on $1-\gamma$ (P3-P4), which means that *examples with anomaly scores close to the decision threshold λ are the ones with a low confidence score* (P5). The next Corollary lists these 5 properties.

Corollary 9. *Given t_1, t_2 as in Theorem 8, the following properties hold for any s, $n = N$, γ, $T \geq 4$:*

- *P1.* $\lim_{N\to+\infty} t_1 = \lim_{N\to+\infty} t_2 = 1 - \gamma$*;*
- *P2. t_1 and t_2 are, respectively, monotonic decreasing and increasing as functions of T;*
- *P3. the interval always contains $1 - \gamma$, i.e. $t_1 \leq 1 - \gamma \leq t_2$;*
- *P4. for $N \to \infty$, there exists s^* with $\psi_N = t^* \in [t_1, t_2]$ such that $t^* \to 1 - \gamma$ and $\mathscr{M}_s \to 0$.*
- *P5. $\psi_N \in [t_1, t_2]$ **iff** $s \in [\lambda - u_1, \lambda + u_2]$, where $u_1(N, \gamma, T), u_2(N, \gamma, T)$ are positive functions.*

Proof sketch. For P1, it is enough to observe that $t_1, t_2 \to 1-\gamma$ for $N \to +\infty$. For P2 and P3, the result comes from simple algebraic steps. P4 follows from the surjectivity of $\mathscr{M}_s$ when $N \to +\infty$, the monotonicity of $\mathbb{P}(\hat{Y}^* = 1|s)$, from P1 with the squeeze theorem. Finally, P5 follows from $\psi_N \in [t_1, t_2] \implies s \in \left[\psi_N^{-1}(t_1), \psi_N^{-1}(t_2)\right]$, as ψ_N is monotonic increasing, where ψ_N^{-1} is the inverse-image of ψ_N. Because for P3 $1-\gamma \in [t_1, t_2]$, it holds that $\psi_N^{-1}(t_1) \leq \psi_N^{-1}(1-\gamma) = \lambda \leq \psi_N^{-1}(t_2)$. This implies that $s \in [\lambda - u_1, \lambda + u_2]$, where $u_1 = \lambda - \psi_N^{-1}(t_1)$, $u_2 = \lambda - \psi_N^{-1}(t_2)$. □

7.1.2 Estimating and Bounding the Rejection Rate

It is important to have an estimate of the rejection rate, which is the proportion of examples for which the model will abstain from making a prediction. This is an important performance characteristic for differentiating among candidate models. Moreover, it is important that not all examples are rejected because such a model is useless in practice. We propose a way to estimate the rejection rate and Theorem 10 shows that our estimate approaches the true rate for large training sets. We strengthen our analysis and introduce an upper bound for the rejection rate, which guarantees that, with arbitrarily high probability, the rejection rate is kept lower than a constant (Theorem 11).

Definition 7 (Rejection rate)**.** *Given the confidence metric $\mathscr{M}_s$ and the rejection threshold τ, the* rejection rate $\mathcal{R} = \mathbb{P}(\mathscr{M}_s \leq \tau)$ *is the probability that a test example with score s gets rejected.*

We propose the following estimator for the reject rate:

Definition 8 (Rejection rate estimator)**.** *Given anomaly scores s with training frequencies ψ_N, let $\phi\colon [0,1] \to [0,1]$ be the function such that $\mathbb{P}(\hat{Y}^* = 1|s) = \phi(\psi_N)$. We define the* rejection rate estimator $\hat{\mathcal{R}}$ *as*

$$\hat{\mathcal{R}} = \hat{F}_{\psi_N}\left(\phi^{-1}\left(1 - e^{-T}\right)\right) - \hat{F}_{\psi_N}\left(\phi^{-1}\left(e^{-T}\right)\right) \tag{7.1}$$

where ϕ^{-1} is the inverse-image through ϕ, and, for $u \in [0,1]$, $\hat{F}_{\psi_N}(u) = \frac{|i \leq N\colon \psi_N(s_i) \leq u|}{N}$ is the empirical cumulative distribution of ψ_N.

Note that $\hat{\mathcal{R}}$ can be computed in practice, as the ψ_N has a distribution that is arbitrarily close to uniform, as stated by Theorem 19 and 20 in the Appendix C.

Theorem 10 (Rejection rate estimate)**.** *Let ϕ be as in Definition 8. Then, for high values of N, $\hat{\mathcal{R}} \approx \mathcal{R}$.*

Proof. From the definition of rejection rate 7, it follows

$$\mathcal{R} = \mathbb{P}\left(\mathscr{M}_s \leq 1 - 2e^{-T}\right) = \mathbb{P}\left(\mathbb{P}(\hat{Y}^* = 1|s) \in \left[e^{-T}, 1-e^{-T}\right]\right) = \mathbb{P}\left(\phi(\psi_N) \in \left[e^{-T}, 1-e^{-T}\right]\right)$$

$$= \mathbb{P}\left(\psi_N \in \left[\phi^{-1}\left(e^{-T}\right), \phi^{-1}\left(1-e^{-T}\right)\right]\right) = F_{\psi_N}\left(\phi^{-1}\left(1-e^{-T}\right)\right) - F_{\psi_N}\left(\phi^{-1}\left(e^{-T}\right)\right).$$

where $F_{\psi_N}(\cdot) = \mathbb{P}(\psi_N \leq \cdot)$ is the theoretical cumulative distribution of ψ_N. Because the true distribution of ψ_N for test examples is unknown, the estimator approximates F_{ψ_N} using the training scores s_i and computes the empirical $\hat{F}_{\psi_N}$. As a result,

$$\mathcal{R} \approx \hat{F}_{\psi_n}\left(g^{-1}\left(1-e^{-T}\right)\right) - \hat{F}_{\psi_n}\left(g^{-1}\left(e^{-T}\right)\right) = \hat{\mathcal{R}}.$$

□

Theorem 11 (Rejection rate upper bound)**.** *Let s be an anomaly score, $\mathscr{M}_s$ be its confidence value, and $\tau = 1 - 2e^{-T}$ be the rejection threshold. For $N \in \mathbb{N}$, $\gamma \in [0, 0.5)$, and small $\delta > 0$, there exists a positive real function $\eta(N, \gamma, T, \delta)$ such that $\mathcal{R} \leq \eta(N, \gamma, T, \delta)$ with probability at least $1 - \delta$, i.e. the rejection rate is bounded.*

Proof. Theorem 8 states that there exists two functions $t_1 = t_1(N, \gamma, T), t_2 = t_2(N, \gamma, T) \in [0, 1]$ such that the confidence is lower than τ if $\psi_N \in [t_1, t_2]$. Moreover, Theorems 19 and 20 claim that ψ_N has a distribution that is close to uniform with high probability (see the theorems and proofs in the Appendix C). As a result, with probability at least $1 - \delta$, we find $\eta(N, \gamma, T, \delta)$ as follows:

$$\mathcal{R} = \mathbb{P}(\mathscr{M}_s \leq 1 - 2e^{-T}) \overbrace{\leq}^{\text{T8}} \mathbb{P}(\psi_N \in [t_1, t_2]) = F_{\psi_N}(t_2) - F_{\psi_N}(t_1)$$

$$\overbrace{\leq}^{\text{T20}} F_{\psi}(t_2) - F_{\psi}(t_1) + 2\sqrt{\frac{\ln\frac{2}{\delta}}{2N}} \overbrace{=}^{\text{T19}} t_2(N, \gamma, T) - t_1(N, \gamma, T) + 2\sqrt{\frac{\ln\frac{2}{\delta}}{2N}} = \eta(N, \gamma, T, \delta).$$

□

7.1.3 Upper Bounding the Expected Test Time Cost

In a learning with reject scenario, there are costs associated with three outcomes: false positives ($c_{fp} > 0$), false negatives ($c_{fn} > 0$), and rejection (c_r) because abstaining typically involves having a person intervene. Estimating an expected per example prediction cost at test time can help with model selection and give a sense of performance. Theorem 12 provides an upper bound on the expected per example cost when (1) using our estimated rejection rate (Theorem 10), and (2) setting the decision threshold λ as in Sec. 2.

Definition 9 (Cost function)**.** *Let Y be the true label random variable. Given the costs $c_{fp}, c_{fn} > 0$, and c_r , the **cost function** is a function $c\colon \{0,1\} \times \{0,1,®\} \to \mathbb{R}$ such that*

$$c(Y, \hat{Y}) = c_r \mathbb{P}(\hat{Y} = ®) + c_{fp}\mathbb{P}(\hat{Y} = 1|Y = 0) + c_{fn}\mathbb{P}(\hat{Y} = 0|Y = 1)$$

Note that defining a specific cost function requires domain knowledge. Following the learning to reject literature, we set an additive cost function. Moreover, the rejection cost needs to satisfy the inequality $c_r \leq \min\{(1-\gamma)c_{fp}, \gamma c_{fn}\}$. This avoids the possibility of predicting always anomaly for an expected cost of $(1-\gamma)c_{fp}$, or always normal with an expected cost of γc_{fn} [184].

Theorem 12. *Let c be a cost function as defined in Definition 9, and ϕ be as in Definition 8. Given a (test) example x with score s, the expected example-wise cost is bounded by*

$$\mathbb{E}_x[c] \leq \min\{\gamma, A\}c_{fn} + (1 - B)c_{fp} + (B - A)c_r, \tag{7.2}$$

where $A = \hat{F}_{\psi_N}(\phi^{-1}\left(e^{-T}\right))$, $B = \hat{F}_{\psi_N}(\phi^{-1}\left(1 - e^{-T}\right))$ are as in Theorem 10.

Proof. We indicate the true label random variable as Y, and the non-rejected false positives and false negatives as, respectively,

$$FP = \mathbb{P}\left(\hat{Y} = 1|Y = 0, \mathscr{M}_s > 1 - 2e^{-T}\right)$$

$$FN = \mathbb{P}\left(\hat{Y} = 0|Y = 1, \mathscr{M}_s > 1 - 2e^{-T}\right)$$

Using Theorem 10 results in

$$\mathbb{E}_x[c] = \mathbb{E}_x[c_{fn}FN + c_{fp}FP + c_r\mathcal{R}] = \mathbb{E}_x[c_{fn}FN] + \mathbb{E}_x[c_{fp}FP] + c_r(B - A)$$

where $A = \hat{F}_{\psi_N}(\phi^{-1}\left(e^{-T}\right))$, $B = \hat{F}_{\psi_N}(\phi^{-1}\left(1 - e^{-T}\right))$ come from Theorem 10. Now we observe that setting a decision threshold λ such that $N \times \gamma$ scores are higher implies that, on expectation, the detector predicts a proportion of positives equal to $\gamma = \mathbb{P}(Y = 1)$. Moreover, for $\varepsilon = 2e^{-T}$,

- $FP \leq \mathbb{P}\left(\hat{Y} = 1|\mathscr{M}_s > 1 - \varepsilon\right) = 1 - B$ as false positives must be less than total accepted positive predictions;
- $FN \leq \gamma$ and $FN \leq \mathbb{P}\left(\hat{Y} = 0|\mathscr{M}_s > 1 - \varepsilon\right) = A$, as you cannot have more false negatives than positives (γ), nor than accepted negative predictions (A).

From these observations, we conclude $\mathbb{E}_x[c] \leq \min\{\gamma, A\}c_{fn} + (1-B)c_{fp} + (B-A)c_r$. □

7.2 Related work

There is no research on learning to reject in unsupervised anomaly detection. However, **three** main research lines are connected to this work.

1) Supervised methods. If some labels are available, one can use traditional supervised approaches to add the reject option into the detector [44, 142]. Commonly, labels can be used to find the optimal rejection threshold in two ways: 1) by trading off the model performance (e.g., AUROC) on the accepted examples with its rejection rate [93, 1], or 2) by minimizing a cost function [169, 37], a risk function [76, 110], or an error function [139, 126]. Alternatively, one can include the reject option in the model and directly optimize it during the learning phase [213, 49, 124].

2) Self-Supervised methods. If labels are not available, one can leverage self-supervised approaches to generate pseudo-labels in order to apply traditional supervised learning to reject methods [107, 210, 77, 141]. For example, one can employ any unsupervised anomaly detector to assign training labels, fit a (semi-)supervised detector (such as DeepSAD [206] or Repen [177]) on the pseudo labels, compute a confidence metric [53], and find the optimal rejection threshold by minimizing the cost function treating the pseudo-labels as the ground truth.

3) Optimizing unsupervised metrics. There exist several unsupervised metrics (i.e., they can be computed without labels) for quantifying detector quality [153], which we described in Section 2. Because they do not need labels, one can find the rejection threshold by maximizing the margin between the detector's quality (computed using such metric) on the accepted and on the rejected examples [192]. This allows us to obtain a model that performs well on the accepted examples and poorly on the rejected ones, which is exactly the same intuition that underlies the supervised approaches.

7.3 Experiments

We experimentally address the following research questions:

Q1. How does RejEx's cost compare to the baselines?

Q2. How does varying the cost function affect the results?

Q3. How does RejEx's CPU time compare to the baselines?

Q4. Do the theoretical results hold in practice?

Q5. Would REJEX's performance significantly improve if it had access to training labels?

7.3.1 Experimental Setup

Methods. We compare **REJEX**[2] against 7 baselines for setting the rejection threshold. These can be divided into three categories: no rejection, self-supervised, and unsupervised metric-based.

We use one method **NOREJ** that always makes predictions and never rejects.

We consider one self-supervised approach **SS-REPEN** [177]. This uses (any) unsupervised detector to obtain pseudo labels for the training set. It then sets the rejection threshold as follows: 1) it creates a held-out validation set (20%), 2) it fits REPEN, a state-of-the-art (semi-)supervised anomaly detector on the training set with the pseudo labels, 3) it computes on the validation set the confidence values as the margin between REPEN's predicted class probabilities $|\mathbb{P}(Y = 1|s) - \mathbb{P}(Y = 0|s)|$, 4) it finds the optimal threshold τ by minimizing the total cost on the validation set.

We consider 5 approaches that employ an existing unsupervised metric to set the rejection threshold and hence do not require labels. **MV** [81], **EM** [81], and **STABILITY** [183] are unsupervised metric-based methods based on stand-alone internal evaluations that use a single anomaly detector to measure its quality, **UDR** [57] and **ENS** [199] are unsupervised consensus-based metrics that an ensemble of detectors (all 12 considered in our experiments) to measure a detector's quality.[3] We apply each of these 5 baselines as follows. 1) We apply the unsupervised detector to assign an anomaly score to each train set example. 2) We convert these scores into class probabilities using [129]. 3) We compute the confidence scores on the training set as difference between these probabilities: $|\mathbb{P}(Y = 1|s) - \mathbb{P}(Y = 0|s)|$. 4) We evaluate possible thresholds on this confidence by computing the unsupervised metric on the accepted and on the rejected examples and select the threshold that maximizes the difference in the metric's value on these two sets of examples. This aligns with the common learning to reject criteria for picking a threshold [40, 192] such that the model performs well on the accepted examples and poorly on the rejected ones.

Data. We carry out our study on 34 publicly available benchmark datasets, widely used in the literature [91]. These datasets cover many application domains, including healthcare (e.g., disease diagnosis), audio and language processing (e.g., speech

[2]Code available at: `https://github.com/Lorenzo-Perini/RejEx`.

[3]Section 2 describes these approaches.

recognition), image processing (e.g., object identification), and finance (e.g., fraud detection). To limit the computational time, we randomly sub-sample 20,000 examples from all large datasets. Table C.1 in the Appendix C provides further details.

Anomaly Detectors and Hyperparameters. We set our tolerance $\varepsilon = 2e^{-T}$ with $T = 32$. Note that the exponential smooths out the effect of $T \geq 4$, which makes setting a different T have little impact. We use a set of 12 unsupervised anomaly detectors implemented in PyOD [258] with default hyperparameters [220] because the unsupervised setting does not allow us to tune them: kNNO [10], IF [151], LOF [29], OCSVM [207], AE [39], HBOS [82], LODA [189], COPOD [145], GMM [4], ECOD [146], KDE [140], INNE [13]. We set all the baselines' rejection threshold via Bayesian Optimization with 50 calls [71].

Setup. For each [dataset, detector] pair, we proceed as follows: (1) we split the dataset into training and test sets (80-20) using 5 fold cross-validation; (2) we use the detector to assign the anomaly scores on the training set; (3) we use either REJEX or a baseline to set the rejection threshold; (4) we measure the total cost on the test set using the given cost function. We carry out a total of $34 \times 12 \times 5 = 2040$ experiments. All experiments were run on an Intel(R) Xeon(R) Silver 4214 CPU with 128GiB of memory.

7.3.2 Experimental Results

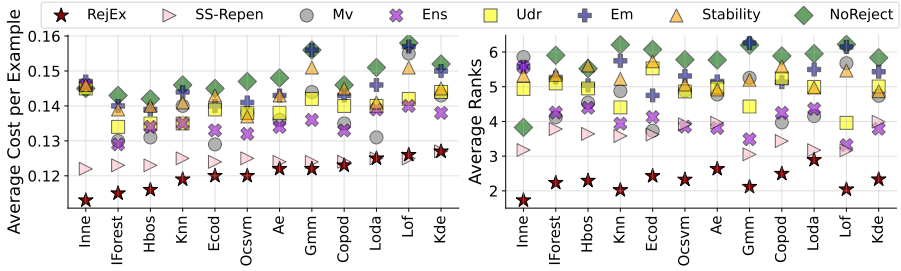

Figure 7.1: Average cost per example (left) and rank (right) aggregated per detector (x-axis) over all the datasets. Our method obtains the lowest (best) cost for 9 out of 12 detectors and it always has the lowest (best) ranking position for $c_{fp} = c_{fn} = 1$, $c_r = \gamma$.

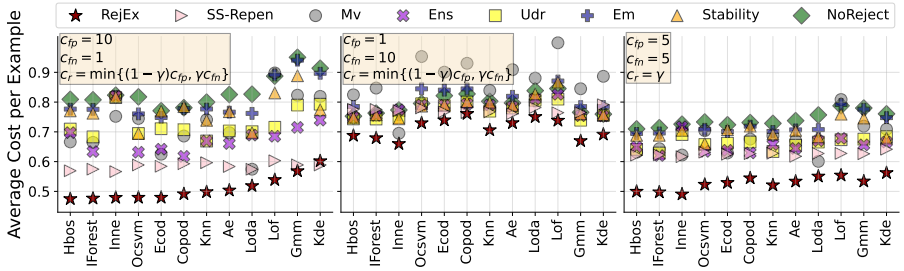

Figure 7.2: Average cost per example aggregated by detector over the 34 datasets when varying the three costs on three representative cases: (left) false positives are penalized more, (center) false negatives are penalized more, (right) rejection has a lower cost than FPs and FNs.

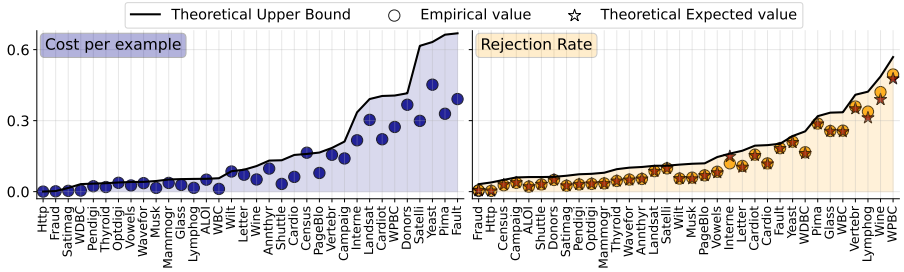

Figure 7.3: Average cost per example (left) and average rejection rate (right) at test time aggregated by dataset over the 12 detectors. In both plots, the empirical value (circle) is always lower than the predicted upper bound (continuous black line), which makes it consistent with the theory. On the right, the expected rejection rates (stars) are almost identical to the empirical values.

Q1: RejEx against the baselines. Figure 7.1 shows the comparison between our method and the baselines, grouped by detector, when setting the costs $c_{fp} = c_{fn} = 1$ and $c_r = \gamma$ (see the Appendix C for further details). RejEx achieves the lowest (best) cost per example for 9 out of 12 detectors (left-hand side) and similar values to SS-Repen when using LODA, LOF and KDE. Averaging over the detectors, RejEx reduces the relative cost by more than 5% vs SS-Repen, 11% vs Ens, 13% vs Mv and Udr, 17% vs Em, 19% vs NoRej. Table C.2 in the Appendix C shows a detailed breakdown.

For each experiment, we rank all the methods from 1 to 8, where position 1 indicates the lowest (best) cost. The right-hand side of Figure 7.1 shows that RejEx always obtains the lowest average ranking. We run a statistical analysis separately for each detector: the Friedman test rejects the null-hypothesis that all methods perform similarly (p-value $< e^{-16}$) for all the detectors. The ranking-based post-hoc Bonferroni-Dunn statistical test [51] with $\alpha = 0.05$ finds that RejEx is significantly better than the baselines for 6 detectors (INNE, IF, HBOS, kNNO, ECOD, OCSVM).

Q2. Varying the costs c_{fp}, c_{fn}, c_r. The three costs c_{fp}, c_{fn}, and c_r are usually set based on domain knowledge: whether to penalize the false positives or the false negatives more depends on the application domain. Moreover, the rejection cost needs to satisfy the constraint $c_r \leq \min\{(1-\gamma)c_{fp}, \gamma c_{fn}\}$ [184]. Therefore, we study their impact on three representative cases: (case 1) high false positive cost ($c_{fp} = 10$, $c_{fn} = 1$, $c_r = \min\{10(1-\gamma), \gamma\}$, (case 2) high false negative cost ($c_{fp} = 1$, $c_{fn} = 10$, $c_r = \min\{(1-\gamma), 10\gamma\}$, and (case 3) same cost for both mispredictions but low rejection cost ($c_{fp} = 5$, $c_{fn} = 5$, $c_r = \gamma$). Note that scaling all the costs has no effect on the relative comparison between the methods, so the last case is equivalent to $c_{fp} = 1$, $c_{fn} = 1$, and $c_r = \gamma/5$.

Figure 7.2 shows results for the three scenarios. Compared to the unsupervised metric-based methods, the left plot shows that our method is clearly the best for high false positives cost: for 11 out of 12 detectors, RejEx obtains both the lowest (or similar for GMM) average cost and the lowest average ranking position. This indicates that using RejEx is suitable when false alarms are expensive. Similarly, the right plot illustrates that RejEx outperforms all the baselines for all the detectors when the rejection cost is low (w.r.t. the false positive and false negative costs). Even when the false negative cost is high (central plot), RejEx obtains the lowest average cost for 11 detectors and has always the lowest average rank per detector. See the Appendix C (Table C.3 and C.4) for more details.

Q3. Comparing the CPU time. Table 7.1 reports CPU time in milliseconds per training example aggregated over the 34 datasets needed for each method to set the rejection threshold on three unsupervised anomaly detectors (IF, HBOS, COPOD). NoRej has CPU time equal to 0 because it does not use any reject option. RejEx takes

Det.	CPU time in ms (mean ± std.)							
	NoRej	**RejEx**	SS-Rep	Mv	Em	Udr	Ens	Stab.
IF	0.0±0.0	**0.06±0.22**	90±68	89±99	155±161	120±132	122±135	916±900
HBOS	0.0±0.0	**0.13±0.93**	89±53	39±81	80±129	200±338	210±358	142±242
COPOD	0.0±0.0	**0.04±0.04**	84±53	21±28	81±60	119±131	123±138	140±248

Table 7.1: Average CPU time (in ms) per training example (± std) to set the rejection threshold aggregated over all the datasets when using IF, HBOS, and COPOD as unsupervised anomaly detector. RejEx has a lower time than all the methods but NoRej, which uses no reject option.

just a little more time than NoRej because computing ExCeeD has linear time while setting a constant threshold has constant time. In contrast, all other methods take 1000× longer because they evaluate multiple thresholds. For some of these (e.g., Stability), this involves an expensive internal procedure.

Q4. Checking on the theoretical results. Section 5.1 introduces three theoretical results: the rejection rate estimate (Theorem 10), and the upper bound for the rejection rate (Theorem 11) and for the cost (Theorem 12). We run experiments to verify whether they hold in practice. Figure 7.3 shows the results aggregated over the detectors. The left-hand side confirms that the prediction cost per example (blue circle) is always $\leq$ than the upper bound (black line). Note that the upper bound is sufficiently strict, as in some cases it equals the empirical cost (e.g., Census, Wilt, Optdigits). The right-hand side shows that our rejection rate estimate (orange star) is almost identical to the empirical rejection rate (orange circle) for most of the datasets, especially the large ones. On the other hand, small datasets have the largest gap, e.g., Wine ($n = 129$), Lymphography ($n = 148$), WPBC ($n = 198$), Vertebral ($n = 240$). Finally, the empirical rejection rate is always lower than the theoretical upper bound (black line), which we compute by using the empirical frequencies ψ_n.

Q5. Impact of training labels on RejEx. We simulate having access to the training labels and include an extra baseline: Oracle uses ExCeeD as a confidence metric and sets the (optimal) rejection threshold by minimizing the cost function using the training labels. Table 7.2 shows the average cost and rejection rates at test time obtained by the two methods. Overall, RejEx obtains an average cost that is only 0.6% higher than Oracle's cost. On a per-detector basis, RejEx obtains a 2.5% higher cost in the worst case (with LODA), while getting only a 0.08% increase in the best case (with KDE). Comparing the rejection rates, RejEx rejects on average only ≈ 1.5 percentage points more examples than Oracle (12.9% vs 11.4%). The Appendix C provides further details.

Table 7.2: Mean ± std. for the **cost per example** (on the left) and the **rejection rate** (on the right) at test time on a per detector basis and aggregated over the datasets.

Detector	Cost per example (Mean±Std.)		Rejection Rate (Mean±Std.)	
	RejEx	Oracle	RejEx	Oracle
Ae	0.126 ± 0.139	0.126 ± 0.139	0.131 ± 0.132	0.118 ± 0.125
COPOD	0.123 ± 0.140	0.121 ± 0.140	0.123 ± 0.131	0.101 ± 0.114
ECOD	0.119 ± 0.138	0.118 ± 0.138	0.125 ± 0.130	0.107 ± 0.114
GMM	0.123 ± 0.135	0.122 ± 0.134	0.139 ± 0.143	0.132 ± 0.136
HBOS	0.118 ± 0.129	0.118 ± 0.129	0.139 ± 0.148	0.114 ± 0.128
IF	0.118 ± 0.129	0.118 ± 0.128	0.127 ± 0.131	0.118 ± 0.130
INNE	0.115 ± 0.129	0.115 ± 0.128	0.132 ± 0.132	0.122 ± 0.125
KDE	0.129 ± 0.140	0.129 ± 0.139	0.121 ± 0.129	0.105 ± 0.120
kNNO	0.119 ± 0.123	0.118 ± 0.123	0.127 ± 0.129	0.112 ± 0.117
LODA	0.125 ± 0.133	0.122 ± 0.130	0.126 ± 0.124	0.110 ± 0.114
LOF	0.126 ± 0.131	0.125 ± 0.131	0.129 ± 0.126	0.118 ± 0.115
OCSVM	0.120 ± 0.131	0.120 ± 0.131	0.126 ± 0.128	0.107 ± 0.115
Avg.	0.122 ± 0.133	0.121 ± 0.133	0.129 ± 0.132	0.114 ± 0.121

Limitations. Because RejEx does not rely on labels, it can only give a coarse-grained view of performance. For example, in many applications anomalies will have varying costs (i.e., there are instance-specific costs) which we cannot account for. Moreover, RejEx has a strictly positive rejection rate, which may increase the cost of a highly accurate detector. However, this happens only in $\approx 5\%$ of our experiments.

7.4 Conclusion

In this Chapter, we focused on unsupervised learning to reject for anomaly detection. The key issue is how to determine the rejection threshold without having access to labels, which all existing methods leverage. To overcome this, we introduced RejEx, which exploits our theoretical analysis of the stability-based confidence metric from ExCeeD to set a constant threshold. Moreover, we proved that a constant rejection threshold comes with strong theoretical guarantees. First, we can estimate the value and an upper bound for the proportion of rejected test examples. Second, we can design a theoretical upper bound on the expected test-time prediction cost per example. In our experimental evaluation, we compared RejEx with several unsupervised metric-based methods and showed that it achieves better costs for the majority of detectors. Additionally, we empirically verified that our theoretical findings hold, and that our estimate of the rejection rate accurately recovers the true value in most cases.

Chapter 8

Conclusions and Future Work

This chapter summarizes the key contributions and provides possible directions for future work.

8.1 Summary

The objective of this dissertation was to push the boundaries of traditional anomaly detection in three unexplored directions.

First, we studied how to transform the anomaly scores into class predictions by setting a decision threshold based on the contamination factor. Because the contamination factor is usually unknown, we propose to estimate it by: (a) acquiring some normal labels, (b) leveraging a related domain with the given contamination, and (c) assuming a Bayesian perspective.

Second, we investigated how to quantify the uncertainty in predictions for unsupervised anomaly detections. That is, we proposed a Bayesian approach to measure a detector's stability when subject to slight modifications of the training set. This new metric does not require any training label, as it targets the detector's consistency of making the same prediction rather than its accuracy.

Third, we explored the practical scenario where unsupervised anomaly detectors need to be deployed and, thus, trusted by practitioners. Because they tend to have high uncertainty in predictions due to the lack of training labels, we proposed to allow the detector to abstain when its prediction has high uncertainty. This has the clear effect of increasing the user trust whenever the detector makes a prediction.

Next, we briefly summarize the main five contributions that enable performing operational, uncertainty-aware and reliable anomaly detection.

8.1.1 Contribution 1: Class prior estimation in active positive and unlabeled learning

We proposed CAPE, a method that estimates the class prior (i.e., one minus the contamination factor) in a PU setting where the positive labels (i.e., normals) were acquired through active learning. CAPE derives the class prior by first estimating each unlabeled example's propensity score, which is the probability that the active learning approach will query the example's label. Theoretically, we proved that our estimate of the class prior will converge to its true value if we obtain accurate propensity scores. Practically, we showed how to estimate the propensity scores in two settings. In the first, the user never makes mistakes and only labels positive examples whereas the second considers modeling the user's uncertainty. Empirically, we demonstrated that CAPE recovers the class prior more accurately than existing approaches.

8.1.2 Contribution 2: Transferring the contamination factor between anomaly detection domains by shape similarity

We proposed a novel method TRADE for estimating the target domain contamination factor given a source dataset with a known contamination factor. The key insight enabling our approach is that the distribution of the normal examples' anomaly scores in both domains will be *similar* if they are derived using the same anomaly detection algorithm. Theoretically, we proved that TRADE's estimate of the contamination factor converges to its actual value when the size of the target dataset increases. Empirically, we demonstrated that TRADE can more accurately estimate the contamination factor than several baselines. More importantly, more accurate estimates lead to improved anomaly detection performance as shown by higher F_1 *scores*.

8.1.3 Contribution 3: Estimating the contamination factor's distribution in unsupervised anomaly detection

We presented γGMM, the first practical method for estimating the posterior distribution of the contamination factor γ in a completely unsupervised manner. We empirically demonstrated on 22 datasets that our mean estimates effectively solve the question of where to threshold the predictions. We outperform all 21 comparison methods and show that the gap in detection accuracy between our estimate and the ground truth (available for these benchmark datasets) is small.

On first impression, the success of our method in solving this challenging and seemingly ill-posed problem may seem surprising. However, it can be attributed to a careful choice of strong inductive biases built into the underlying probabilistic model. We argue that all of the following elements are necessary, each substantially contributing to the overall success: (i) representing the data in the space of anomaly detector scores defines a meaning for the dimensions and allows borrowing inductive biases of arbitrary detector algorithms, (ii) the mixture model encodes a natural clustering assumption for both the normal samples and the anomalies, (iii) the ordering used for determining the final distribution incorporates both the location and shape of the mixture components in a carefully balanced manner and (iv) the transformation from the ordering to probabilities is robustly parameterized via just two intuitive hyperparameters, enabling the use of the same defaults for all cases.

8.1.4 Contribution 4: A theoretical framework for assessing an anomaly detector's example-wise stability

We proposed a method to estimate the stability of anomaly detectors in their example-wise class predictions. We first formally defined stability as the probability that example-wise predictions change due to perturbations in the training set. Then, we introduced ExCeeD, a method that estimates the stability using a two-step approach. First, we estimate smooth outlier probabilities using a Bayesian approach. Second, we use the estimated outlier probabilities to derive a stability score on an example-by-example basis. A large experimental comparison shows that our approach can recover stability scores matching empirical frequencies.

8.1.5 Contribution 5: Unsupervised Anomaly Detection with Rejection

We addressed learning to reject in the context of unsupervised anomaly detection. The key challenge was how to set the rejection threshold without access to labels which are required by all existing approaches We proposed an approach RejEx that exploits our theoretical analysis of the ExCeeD stability metric. Our new analysis shows that it is possible to set a constant rejection threshold and that doing so offers strong theoretical guarantees. First, we can estimate the proportion of rejected test examples and provide an upper bound for our estimate. Second, we can provide a theoretical upper bound on the expected test-time prediction cost per example. Experimentally, we compared RejEx against several (unsupervised) metric-based methods and showed that, for the majority of anomaly detectors, it obtained lower (better) cost. Moreover, we proved that our theoretical results hold in practice and that our rejection rate estimate is almost identical to the true value in the majority of cases.

8.2 Future Research Directions

Improving the example-wise stability metric. Chapter 6 introduced a stability metric that quantifies the anomaly detector's consistency in making the same prediction, given a value for the contamination factor. However, this metric can be substantially improved in two complementary directions. First, deterministic estimates of the contamination factor may yield sub-optimal stability values. Thus, a question is: *how can we design a stability metric that uses a (posterior) distribution of contamination factor values?* Second, one usually tests a model on batches of data (i.e., the test set has more than one sample). *Can we design a batch-wise stability metric that leverages the properties of a batch of examples?*

Active learning strategy for models with rejection. In anomaly detection, a current trend involves weakly-supervised approaches using active learning to gather a few targeted labels. However, there is a challenge in developing an active learning strategy for a model with rejection. The reject option is designed to reject ambiguous examples near the decision boundary, while active learning refines the decision boundary by selecting examples close by. This creates a potential issue where typical active learning strategies may gather sub-optimal labels by querying areas the model would reject during testing. The central question is: *How do we design an optimal active learning strategy for a model with rejection?*

Calibration with flexible supervision. Transforming scores into calibrated probabilities requires ground truth labels because it needs to link the estimated probabilities to the true class frequencies. Although the current literature only explores fully supervised settings, weakly supervised methods may benefit from having calibrated probabilities. In scenarios like anomaly detection, where collecting labels is challenging, adapting calibration techniques to leverage limited labeled examples and a majority of unlabeled examples can potentially increase the accuracy and reliability of the model. The central question is: *Can we develop algorithms to calibrate the detector's scores using both labeled and unlabeled examples?*

Evaluation metric for decision threshold. Evaluating the quality of a decision threshold is a challenging task. In this dissertation, we proposed to evaluate the quality of the decision threshold by computing the F_1 score, a suitable metric for anomaly detection. However, because the F_1 score targets the quality of the positive class predictions, it might lead to a biased evaluation of the decision threshold. For instance, a detector that flips the anomaly score ranking would have an F_1 score equal to 0 when using the true contamination factor, and an F_1 score equal to $\frac{2\gamma}{1+\gamma} > 0$ when always predicting anomaly class. This raises the question: *Can we design a proper evaluation metric to assess the quality of a chosen decision threshold?*

Anomaly augmentation for tabular data using foundation models. Most anomaly detection setting includes a tiny proportion of anomalies because anomalous examples are often hard to collect. However, it has been shown that supervised/semi-supervised detectors obtain higher performance when trained on a large set of anomalies. Recently, foundation models have become important and accurate in generating realistic fake examples, particularly for image data. This opens up a new question: *How can we leverage foundation models to generate realistic anomalies to learn an anomaly detector on a broader set of anomalies?*

Data quality metric for fake anomalies. Assuming that a generator for anomalies exists, it is quite likely that not all of the generated examples are useful for training an anomaly detection model. For instance, some fake anomalies may be unrealistic, i.e. they can never occur in the real setting, and including them in the training set may deteriorate the detector's performance. Even worse, some fake anomalies may be indistinguishable from normal patterns, i.e. they overlap with normals, and learning a detector from mislabeled examples could further deteriorate its performance. Thus, the central question is: *How do we design a data quality metric for fake anomalous examples, where high-value fakes would be included in the training set?*

Appendix A

Transferring the contamination factor between anomaly detection domains by shape similarity

This Appendix refers to Chapter 4 and contains the full proofs for our theoretical results and additional details about our experiments:

- First, we provide the proofs for the two propositions listed in the methodology section (Sec. 4.1).
- Second, we give the proof of our Theorem 5 presented in the theoretical analysis section. To do so, we provide a number of sub-theorems that are needed to prove the main result.
- Finally, we further supply additional details about our experiments and results.

A.1 Proofs of methodology section propositions

Proof of Proposition 3. By definition, u^{λ} is a probability density function if two properties hold:

1. *Non-Negativity.* Since $u(x)$ is always non-negative, it follows that $u^\lambda(x) \geq 0$ for any $x \in [0, 1]$;

2. *Integral equal to* 1. Integrating on the support,

$$\int_0^1 u^\lambda(x)\,dx = \int_0^1 \frac{\lambda v(\lambda x)}{\int_0^\lambda v(x')\,dx'}\,dx = \int_0^\lambda \frac{\cancel{\lambda} v(z)}{\cancel{\lambda}\int_0^\lambda v(x')\,dx'}\,dz = 1,$$

because the same integral is on both sides of the ratio. □

Proof of Proposition 4. By hypothesis, $\gamma_m^T = \mathbb{P}\left(T_m \geq \lambda_m^T\right) = \mathbb{P}\left(Y_m^T = 1\right)$. Then,

$$\mathbb{E}\left[\hat{\gamma}_m^T\right] = \mathbb{E}\left[\frac{\sum_{i=1}^m \mathbb{1}_{\{f(x)\geq\lambda_m^T\}}(x_i)}{m}\right] = \sum_{i=1}^m \frac{\mathbb{E}\left[\mathbb{1}_{\{f(x)\geq\lambda_m^T\}}(x_i)\right]}{m}$$

$$= \sum_{i=1}^m \frac{\mathbb{P}\left(\left\{f(x_i) \geq \lambda_m^T\right\}\right)}{m} = \sum_{i=1}^m \frac{\mathbb{P}\left(T_m \geq \lambda_m^T\right)}{m} = \sum_{i=1}^m \frac{\gamma_m^T}{m} = \gamma_m^T,$$

where the equality $\mathbb{E}[\mathbb{1}_{\{f(x)\geq\lambda_m^T\}}(x)] = \mathbb{P}(\{f(x) \geq \lambda_m^T\})$ holds by the definition of indicator function.

A.2 Theoretical Convergence Analysis

Our **main theoretical result** can be summarized as follows. For $m \to +\infty$,

- *Our estimate of the **target threshold** λ_m^T converges to the real target value λ^T (Theorem 5 first part);*
- *Our estimate of the **target contamination factor** $\hat{\gamma}_m^T$ converges on average to the real target value γ^T (Theorem 5 second part).*

Because the contamination factor is derived from the predictive threshold, we first focus on proving that the target predictive threshold of T_m converges to the actual value of the ground-truth T. Essentially, the equalities in Eq. 4.4 hold if the limit symbol is allowed to pass out through the functions. We motivate the four steps of Eq. 4.4 by answering the four following questions:

Q1. Does the set of values minimizing $KL(S^{\lambda^S} \parallel T^\lambda)$ contains only the real source threshold λ^T?

Q2. Given that $t_m \to t$ uniformly, does it hold for any λcut distribution, i.e. $t_m^\lambda \to t^\lambda$ for any $\lambda \in [\delta, 1]$?

Q3. Does the KL divergence $KL(S^{\lambda^S} \parallel T_m^\lambda)$ converge to the KL divergence $KL(S^{\lambda^S} \| T^\lambda)$?

Q4. Are the function arg min and the limit interchangeable?

Finally, we move to the contamination factor by answering to:

Q5. Does the target contamination factor's estimate $\hat{\gamma}_m^T$ converge to the true target contamination factor γ^T?

Q1. Uniqueness of the Limit

Given that the optimization problem in Eq. 4.1 may return a set of solutions, we first investigate the uniqueness of the theoretical target threshold λ^T, which is obtained as a solution of the optimization problem with the real T instead of T_m. For this task, we take advantage of the following theorem stating that if two λcut distributions are equal *almost surely*, then the two thresholds must be equal.

Theorem 13. *Let T be a real continuous random variable with probability density function $t\colon [0,1] \to (0, +\infty)$. Let's assume that there exists $\lambda_1,\ \lambda_2 \in [\delta, 1]$, for any fixed $\delta > 0$, such that the equality*

$$t^{\lambda_1}(x) = t^{\lambda_2}(x)$$

holds for almost every $x \in [0,1]$, where t^{λ_1} and t^{λ_2} are, respectively, the λ_1 and λ_2 cut distributions of t. Then,

$$\lambda_1 = \lambda_2.$$

Proof. Since $\lambda_1,\ \lambda_2 \in [\delta, 1]$, there exists a value $a \in [\max\{\delta - 1, -\lambda_1, -\lambda_2\}, 1 - \delta]$ such that $\lambda_1 = \lambda_2 + a$. Then, for almost every $x \in [0,1]$,

$$t^{\lambda_1}(x) = t(\lambda_1 x) \cdot \frac{\lambda_1}{\int_0^{\lambda_1} t(x')\,dx'} = t((\lambda_2 + a)x) \cdot \frac{\lambda_2 + a}{\int_0^{\lambda_2 + a} t(x')\,dx'}$$

and

$$1 = \frac{t^{\lambda_1}(x)}{t^{\lambda_2}(x)} = \frac{t((\lambda_2 + a)x)}{t(\lambda_2 x)} \cdot \frac{(\lambda_2 + a)\int_0^{\lambda_2} t(x')\,dx'}{\lambda_2 \int_0^{\lambda_2 + a} t(x')\,dx'} = \frac{t((\lambda_2 + a)x)}{t(\lambda_2 x)} \cdot \frac{1}{c},$$

where $\frac{1}{c} > 0$ refers to the second factor. Hence, we derive that

$$t((\lambda_2 + a)x) = c \cdot t(\lambda_2 x) \implies t(bz) = c \cdot t(z),$$

where $b = \left(1 + \frac{a}{\lambda_2}\right) \in \left(\max\left\{0, 1 - \frac{1-\delta}{\lambda_1}, 1 - \frac{\lambda_2}{\lambda_1}\right\}, 1 + \frac{1-\delta}{\lambda_1}\right)$ and $z = \lambda_2 x \in [0, \lambda_2]$. By recurrence, for any $n \in \mathbb{N}$,

$$t(z) = c \cdot t\left(\frac{z}{b}\right) = \cdots = c^n \cdot t\left(\frac{z}{b^n}\right).$$

If $a > 0$, then it is easy to check that

$$c = \frac{\lambda_2}{\lambda_2 + a} \cdot \frac{\int_0^{\lambda_2 + a} s(z)\, dz}{\int_0^{\lambda_2} t(z)\, dz} < 1.$$

Then, for $n \to +\infty$

$$t(z) = c^n \cdot t\left(\frac{z}{b^n}\right) \to 0 \text{ for all } y \in [0, \lambda_2],$$

which means that $t(z) = 0$ for all $y \in [0, \lambda_2]$ and, consequently, $t(x)$ is constant and equal to 0 for all $x \in [0, 1]$. This is a contradiction.

On the other hand, if $a < 0$ then $c > 1$. However, because t is positive and, especially, $t(0) > 0$, we can conclude that

$$t(z) = c^n \cdot t\left(\frac{z}{b^n}\right) \to +\infty \implies t(z) = +\infty \quad \forall\, z \in [0, \lambda_2],$$

which is again a contradiction. Thus, $a = 0$ and $\lambda_1 = \lambda_2$. □

We exploit this result to show the uniqueness of the limit solution.

Theorem 14 (Uniqueness of the solution)**.** *Let S and T be two real continuous random variables with density, respectively, $s, t\colon [0, 1] \to (0, +\infty)$. Let $\lambda^S, \lambda^T \in [\delta, 1]$ be two fixed thresholds, for any $\delta > 0$, such that*

$$KL\left(S^{\lambda^S} \,\middle\|\, T^{\lambda^T}\right) = 0, \tag{A.1}$$

where S^{λ^S}, T^{λ^T} are, respectively, the λ^S and λ^T cut distributions. Then,

$$\arg\min_{\lambda \in [\delta, 1]} \left\{KL\left(S^{\lambda^S} \middle\| T^{\lambda}\right)\right\} = \lambda^T.$$

Proof. By the Gibbs inequality, for any $\lambda \in [\delta, 1]$, the inequality

$$KL\left(S^{\lambda^S} \,\middle\|\, T^{\lambda}\right) \geq 0$$

holds. Thus, the inclusion

$$\lambda^T \in \arg\min_{\lambda \in [\delta, 1]} \left\{KL\left(S^{\lambda^S} \,\middle\|\, T^{\lambda}\right)\right\}$$

comes directly from the hypothesis. Let's now assume that there exists another global minimum $\lambda^* \in [\delta, 1]$ such that

$$\lambda^* \in \arg\min_{\lambda \in [\delta,1]} \left\{ KL\left(S^{\lambda^S} \,\middle\|\, T^{\lambda}\right) \right\},$$

which implies

$$KL\left(S^{\lambda^S} \,\middle\|\, T^{\lambda^*}\right) = 0.$$

We now prove that, for almost every $x \in [0, 1]$, $t^{\lambda^T}(x) = t^{\lambda^*}(x)$. Let's start from $t^{\lambda^*}(x)$. By definition of KL divergence,

$$\overbrace{KL\left(S^{\lambda^S} \,\middle\|\, T^{\lambda^*}\right)}^{=0} = \int_0^1 s^{\lambda^S}(x) \log\left(\frac{s^{\lambda^S}(x)}{t^{\lambda^*}(x)}\right) dx$$

$$= -\int_0^1 s^{\lambda^S}(x) \log\left(\frac{t^{\lambda^*}(x)}{s^{\lambda^S}(x)}\right) dx \geq -\int_0^1 s^{\lambda^S}(x) \left(\frac{t^{\lambda^*}(x)}{s^{\lambda^S}(x)} - 1\right) dx = -\overbrace{\int_0^1 \left(t^{\lambda^*}(x) - s^{\lambda^S}(x)\right) dx}^{=0}$$

where the only inequality comes from the property of logarithms $\log(x) \leq x - 1$ for $x \in (0, 1]$, and the final result is 0 because each integral is equal to 1. As a result, the inequality turns out to be an equality and, for almost every $x \in [0, 1]$, $s^{\lambda^S}(x) = t^{\lambda^*}(x)$. By repeating the same procedure with $t^{\lambda^T}(x)$ instead of $t^{\lambda^*}(x)$, we get that, for almost every $x \in [0, 1]$, $s^{\lambda^S}(x) = t^{\lambda^T}(x)$. Because the union of two sets with measure equal to 0 is still a set with measure equal to 0 and by transitivity, we conclude that $t^{\lambda^*}(x) = t^{\lambda^T}(x)$ for almost every $x \in [0, 1]$. By applying Theorem 13 to t^{λ^*} and t^{λ^T}, we prove that $\lambda^* = \lambda^T$, meaning that the solution is unique. □

Theorem 14 proves that only one solution exists, so that if the sequence of estimated predictive thresholds λ_m^T converges, then its limit is exactly the theoretical predictive threshold λ^T.

Q2. Convergence of λcut distributions

Given that the λcut distribution is an actual probability density function (Prop. 3), the following theorem shows that assuming that the target density t_m converges uniformly to the theoretical density t, then the same relationship holds for any of their λcut distributions.

Theorem 15 (Uniform convergence of λcut distributions)**.** *Let t_m, for any $m \in \mathbb{N}$, be a sequence of continuous probability density functions such that $t_m \to t$ uniformly in*

$[0, 1]$ *for* $m \to +\infty$. *Then, for any* $\lambda \in [\delta, 1]$,

$$t_m^\lambda \xrightarrow{m\to+\infty} t^\lambda \quad \textit{uniformly in } [0, 1],$$

where t_m^λ *and* t^λ *are* λ*cut distributions as defined in definition 4.*

Proof. Let's fix $\varepsilon > 0$ and $\lambda \in [\delta, 1]$. Since t is bounded, there exists a constant $K \geq 0$ such that $|t(x)| \leq K$ for all $x \in [0, 1]$. As $t_m \to t$ uniformly, there exists $M_1 \in \mathbb{N}$ such that

$$|t_m(x) - t(x)| < 1 \quad \forall\, x \in [0, 1],\ m \geq M_1,$$

which is equivalent to

$$|t_m(x)| < |t(x)| + 1 \leq K + 1 \quad \forall\, x \in [0, 1],\ m \geq M_1.$$

Because of the uniform convergence of t_m, which is continuous, bounded and with integral equal to 1, the sequence $\int_0^\lambda t_m(z)\,dz$ converges to $\int_0^\lambda t(z)\,dz$ and so does the reciprocal sequence.

Thus, by definition, there exists $M_2 \in \mathbb{N}$ such that

$$\left| \frac{1}{\int_0^\lambda t_m(z)\,dz} - \frac{1}{\int_0^\lambda t(z)\,dz} \right| < \frac{\varepsilon}{2\lambda(K+1)} \quad \forall\, m \geq M_2.$$

Likewise, given that t_m converges uniformly to t, there exists $M_3 \in \mathbb{N}$ such that

$$|t_m(x) - t(x)| < \frac{\varepsilon \int_0^\lambda t(z)\,dz}{2\lambda} \quad \forall\, x \in [0, 1],\ m \geq M_3.$$

Let's indicate by $M = \max\{M_1, M_2, M_3\}$. For $m \geq M$ and $x \in [0, 1]$,

$$\left|t_m^\lambda(x) - t^\lambda(x)\right| = \lambda \left| \frac{t_m(x)}{\int_0^\lambda t_m(z)dz} - \frac{t(x)}{\int_0^\lambda t(z)dz} \right| \leq \left| \frac{\lambda t_m(x)}{\int_0^\lambda t_m(z)dz} - \frac{\lambda t_m(x)}{\int_0^\lambda t(z)dz} \right| + \left| \frac{\lambda t_m(x)}{\int_0^\lambda t(z)dz} - \frac{\lambda t(x)}{\int_0^\lambda t(z)dz} \right|$$

$$\leq \lambda |t_m(x)| \left| \frac{1}{\int_0^\lambda t_m(z)dz} - \frac{1}{\int_0^\lambda t(z)dz} \right| + \lambda \frac{|t_m(x) - t(x)|}{\int_0^\lambda t(z)dz} \leq \frac{\lambda(K+1)\varepsilon}{2\lambda(K+1)} + \frac{\lambda\varepsilon \int_0^\lambda t(z)dz}{2\lambda \int_0^\lambda t(z)dz} = \varepsilon,$$

where in the last step we exploit the previous inequalities. As a result, $t_m^\lambda \to t^\lambda$ uniformly for any $\lambda \in [\delta, 1]$ when $m \to +\infty$. □

Theorem 15 guarantees that the uniform convergence of the full density t_m to t (Assumption 1) is enough to claim that any λcut distribution of t_m converges to the corresponding λcut distribution of t.

Q3. Convergence of KL Divergence

Given the uniform convergence of the λcut distributions, we still need to verify whether their KL divergence converges, as expected, to the KL divergence of their limit. The following theorem shows that the KL divergence and the limit symbol can be interchanged.

Theorem 16 (Interchangeability of KL divergence and limit symbol)**.** *Let S, T and T_m be three real continuous random variables with probability density functions, respectively, s, t, $t_m\colon [0,1] \to (0,+\infty)$. Fix $\lambda^S, \lambda \in [\delta, 1]$, with $\delta > 0$. Let s^{λ^S}, t^λ and t_m^λ be λcut distributions. Assume that $s^{\lambda^S}(x)$, $t^\lambda(x)$, $t_m^\lambda(x) > 0$ for all $x \in [0,1]$, and that $t_m^\lambda \to t^\lambda$ uniformly for all $x \in [0,1]$. Then,*

$$\lim_{m\to+\infty} KL\left(S^{\lambda^S} \,\middle\|\, T_m^\lambda\right) = KL\left(S^{\lambda^S} \,\middle\|\, \lim_{m\to+\infty} T_m^\lambda\right) = KL\left(S^{\lambda^S} \,\middle\|\, T^\lambda\right),$$

where S^{λ^S}, T^λ and T_m^λ are the unique variables with, respectively, s^{λ^S}, t^λ and t_m^λ as probability density functions.

Proof. By definition,

$$KL\left(S^{\lambda^S} \,\middle\|\, T_m^\lambda\right) = \int_0^1 s^{\lambda^S}(x) \log\left(\frac{s^{\lambda^S}(x)}{t_m^\lambda(x)}\right) dx. \tag{A.2}$$

Because it is both positive and continuous, t_m^λ takes its (positive) maximum and minimum values on the interval $[0,1]$ and, therefore, the reciprocal function is bounded. Following the same procedure as in Theorem 15, it is easy to prove that

$$\frac{s^{\lambda^S}(x)}{t_m^\lambda(x)} \to \frac{s^{\lambda^S}(x)}{t^\lambda(x)} \quad \text{uniformly for } x \in [0,1].$$

Likewise, by exploiting the continuity of the logarithm function and both the continuity and the boundedness of the λcut distributions, the following convergence naturally holds for $x \in [0,1]$

$$s^{\lambda^S}(x) \log\left(\frac{s^{\lambda^S}(x)}{t_m^\lambda(x)}\right) \to s^{\lambda^S}(x) \log\left(\frac{s^{\lambda^S}(x)}{t^\lambda(x)}\right) \text{ uniformly.}$$

Let's now fix $\varepsilon > 0$. By definition of uniform convergence, there exists $M \in \mathbb{N}$ such that, for all $m \geq M$ and for any $x \in [0,1]$,

$$\left| s^{\lambda^S}(x) \log\left(\frac{s^{\lambda^S}(x)}{t_m^\lambda(x)}\right) - s^{\lambda^S}(x) \log\left(\frac{s^{\lambda^S}(x)}{t^\lambda(x)}\right) \right| < \varepsilon.$$

Consequently, for all $m \geq M$,

$$\left| KL\left(S^{\lambda^S} \middle\| T_m^\lambda\right) - KL\left(S^{\lambda^S} \middle\| T_m^\lambda\right) \right| = \left| \int_0^1 s^{\lambda^S}(x) \log\left(\frac{s^{\lambda^S}(x)}{t_m^\lambda(x)}\right) dx - \int_0^1 s^{\lambda^S}(x) \log\left(\frac{s^{\lambda^S}(x)}{t^\lambda(x)}\right) dx \right|$$

$$\leq \int_0^1 \left| s^{\lambda^S}(x) \log\left(\frac{s^{\lambda^S}(x)}{t_m^\lambda(x)}\right) - s^{\lambda^S}(x) \log\left(\frac{s^{\lambda^S}(x)}{t^\lambda(x)}\right) \right| dx \leq \varepsilon,$$

which proves the thesis. □

Provided that the limit can pass outside the KL divergence function, next we discuss the convergence of the transfer predictive threshold when solving the optimization problem at each step $m \in \mathbb{N}$.

Q4. Interchangeability of arg min and limit

The final step to prove that the transfer threshold λ_m^T converges to the actual value λ^T (first main result in Eq. 4.4) consists of exchanging the limit symbol with the argmin function. Although in general the equality may not hold, in our scenario we only need to show that one of the two inequalities is true. If we proved that

$$\limsup_{m \to +\infty} \operatorname*{arg\,min}_{\lambda \in [\delta, 1]} \left\{ KL\left(S^{\lambda^S} \middle\| T_m^\lambda\right) \right\} \subseteq \operatorname*{arg\,min}_{\lambda \in [\delta, 1]} \left\{ \lim_{m \to +\infty} KL\left(S^{\lambda^S} \middle\| T_m^\lambda\right) \right\}$$

we could conclude through Theorem 14 that the right-hand side contains only one solution and, in turn, that the equality holds. For this task, we first need to introduce the following theorem, which states that the KL divergence is a continuous function when the threshold λ varies in $[\delta, 1]$.

Theorem 17 (Continuity of KL divergence)**.** *Given two continuous random variables S and T_m with probability density functions, respectively, $s(x)$ and $t_m(x)$ defined on $[0, 1]$, and a constant $\delta > 0$, then the function*

$$g_m(\lambda) := KL\left(S^{\lambda^S} \middle\| T_m^\lambda\right) \text{ is continuous in } [\delta, 1],$$

where S^{λ^S} and T_m^λ are the random variable with, respectively, λcut distribution equal to s^{λ^S} and t_m^λ, as defined in definition 4.

Proof. By definition, for any $\lambda \in [\delta, 1]$,

$$t_m^\lambda(x) = t_m(\lambda x) \cdot \frac{\lambda}{\int_0^\lambda t_m(z)\, dz}.$$

The reciprocal function of the integral depends on λ and is continuous because it is bounded and t_m is continuous with respect to both x and λ. Because the product of continuous functions is still a continuous function, $t_m^\lambda(x)$ is continuous on $[\delta, 1]$. As a result,

$$g_m(\lambda) = \int_0^1 s^{\lambda^S}(x) \log\left(\frac{s^{\lambda^S}(x)}{t_m^\lambda(x)}\right) dx = \int_0^1 s^{\lambda^S}(x)\Big[\log\left(s^{\lambda^S}(x)\right) - \log\left(t_m^\lambda(x)\right)\Big] dx$$

$$= \int_0^1 s^{\lambda^S}(x) \log\left(s^{\lambda^S}(x)\right) dx - \int_0^1 s^{\lambda^S}(x) \log\left(t_m^\lambda(x)\right) dx$$

is continuous on $[\delta, 1]$ because of combination of operations that preserve the continuity. In the last line, only the second integral depends on λ through the function $t_m^\lambda(x)$, hence preserving the continuity of the integrand function. Thus, $g_m(\lambda)$ is continuous. □

This guarantees that the hypotheses of the following theorem hold.

Theorem 18 (Interchangeability of arg min and limit symbol)**.** *Let $g_m\colon [\delta, 1] \to \mathbb{R}$ be a continuous sequence of functions for a fixed $\delta > 0$. Assume that g_m converges pointwise to a continuous function $g\colon [\delta, 1] \to \mathbb{R}$. Then,*

$$\limsup_{m\to+\infty}\left(\arg\min_{\lambda\in[\delta,1]} g_m(\lambda)\right) \subseteq \arg\min_{\lambda\in[\delta,1]}\left(\lim_{m\to+\infty} g_m(\lambda)\right) = \arg\min_{\lambda\in[\delta,1]} g(\lambda).$$

Proof. To prove the result, we take an element in the set on the left-hand side and show that the element belongs to the set on the right-hand side. Let $\bar{\lambda} \in [\delta, 1]$ be such that

$$\bar{\lambda} \in \limsup_{m\to+\infty}\left(\arg\min_{\lambda\in[\delta,1]} g_m(\lambda)\right).$$

Then, by definition of limit superior, for all $m \in \mathbb{N}$, $\exists\, K(m) \in \mathbb{N}$, $K(m) \geq m$, such that $\bar{\lambda} \in \arg\min_{\lambda\in[\delta,1]} g_{K(m)}(\lambda)$. For $\lambda \in [\delta, 1]$,

$$g_{K(m)}(\bar{\lambda}) \leq g_{K(m)}(\lambda).$$

Given that $\lim_{m\to+\infty} K(m) = +\infty$, the inequality

$$g(\bar{\lambda}) = \lim_{m\to+\infty} g_{K(m)}(\bar{\lambda}) \leq \lim_{m\to+\infty} g_{K(m)}(\lambda) = g(\lambda)$$

holds for all $\lambda \in [\delta, 1]$. As a result, $\bar{\lambda} \in \arg\min_{\lambda\in[\delta,1]} g(\lambda)$. □

We apply this theorem by using $g_m(\lambda) = KL(S^{\lambda^S} \| T_m^\lambda)$, which is continuous according to Theorem 17. In addition, the pointwise convergence to $g(\lambda) = KL(S^{\lambda^S} \| T^\lambda)$ holds by the Theorem 16.

A.3 Experiments

Table A.1: The number of examples, variables, and the contamination factor for each considered dataset. IoT datasets (last 9) contamination factor vary among a list of values.

Dataset	N	d	γ	Dataset	N	d	γ
store1-hour1	1703	11	0.1274	turbine-15	838	10	0.101
store1-hour2	1703	11	0.1192	turbine-21	385	10	0.078
store1-hour3	1703	11	0.1262	Webcam	2000	115	List
store1-hour4	1703	11	0.1315	Security Camera 838	2000	115	List
store2-hour1	1704	11	0.0370	Security Camera 737	2000	115	List
store2-hour2	1704	11	0.0082	Security Camera 1003	2000	115	List
store2-hour3	1704	11	0.0540	Security Camera 1002	2000	115	List
store2-hour4	1704	11	0.0769	Ennio Doorbell	2000	115	List
store3-hour1	1276	11	0.0337	Ecobee Thermostat	2000	115	List
store3-hour2	1276	11	0.0713	Danmini Doorbell	2000	115	List
store3-hour3	1276	11	0.0165	Baby Monitor	2000	115	List
store3-hour4	1276	11	0.0681				

Table A.2: Comparison of the performance of TRADE, which uses an ensemble of detectors to estimate the contamination factor, to a variant that only uses a single detector. Performance is measured in terms of the accuracy of the estimate of γ as measured by mean absolute error (MAE). We report the number of times TRADE wins (lower MAE), draws (absolute differences ≤ 0.001), and loses (higher MAE) versus each variant. Each variant is identified by the name of the considered anomaly detector.

TRADE Variant	**MAE on γ** Wins	Draws	Loses
KNNO	**105**	2	27
IF	**74**	5	55
CBLOF	**91**	5	38
COPOD	**95**	3	36
LODA	**85**	4	45
VAE	**101**	1	32
HBOS	**113**	1	20
SOD	**101**	9	24
LSCP	**82**	0	52
Total	**847**	30	329

Data. Table A.1 shows the properties of the 23 real-world anomaly detection datasets. The water data are proprietary and shared with the researchers under an NDA. They

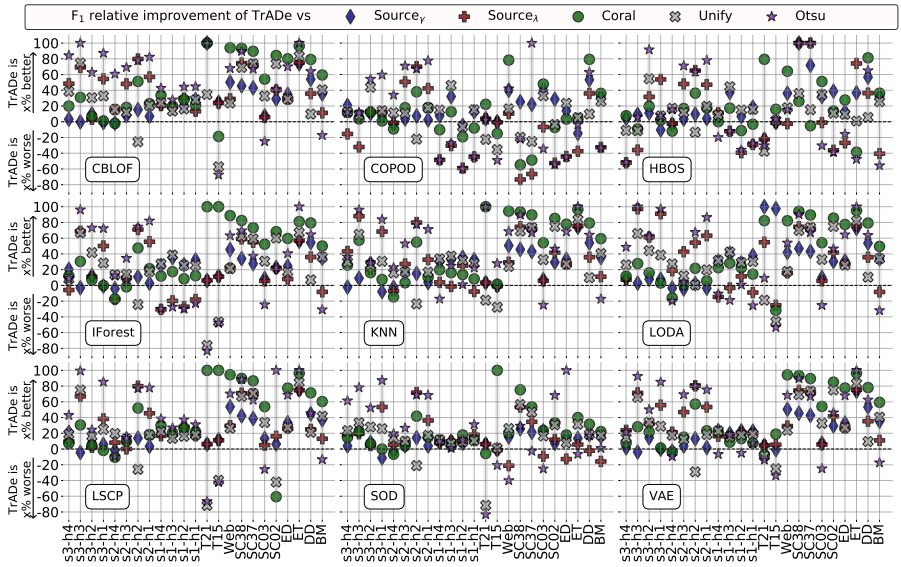

Figure A.1: F_1 *relative improvement* of TrADe over the baselines, divided by anomaly detector h and averaged per target domain, over all the 206 experiments. Positive values mean that TrADe performs better than the baseline. Overall, TrADe shows positive F_1 scores improvements against all the baselines in at least 16 target domains for 7 out of 9 anomaly detectors. However, when averaging over the target domains, TrADe performs worse than Source$_\lambda$ only when COPOD is used as an anomaly detector.

cannot be made public without consent of the providing company. The wind turbine data can be downloaded from `http://www.industrial-bigdata.com/Data` [254]. The IoT data can be found at `https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT#` [162, 164]. While water and wind turbines data preserve their original structure, we subsample 2000 examples from the IoT data setting the contamination factor as $[0.03, 0.05, 0.08, 0.10, 0.15, 0.20, 0.25]$ when used as source domain, and as 0.01 when used as target domain.

Hyperparameters. Because our goals are i) to recover the contamination factor, and ii) prove the impact of more accurate estimates on the anomaly detector's performance, we select sensible hyperparameters and set the remaining as default[1]: kNNO has $k = 5$, CBLOF has $n\_clusters = 5$, HBOS has $n\_bins = 5$, SOD has $n\_neighbors = 10$ and $ref\_set = 5$, IF has $n\_estimators = 5$, LODA has $n\_bins = 5$ and $n\_random\_cuts =$

[1] implementation is available on PyOD: `https://pyod.readthedocs.io/en/latest/`

100, LSCP has *local_region_size* = 20 and uses 3 LOF with $k = 20$, VAE has [30, 6, 6, 30] as layer structure and *epochs* = 10, while COPOD has no hyperparameters.

Results (Q2). Figure A.1 shows the F_1 *relative improvement* for each anomaly detector h over all the 206 experiments, averaged by target domain. For 7 out of 9 anomaly detectors, TRADE performs on average better than each baseline in at least 16 out of 23 target domains. When using HBOS as anomaly detector, TRADE has an average improvement between 11.5% and 21% against all the baselines, despite showing negative improvements in 13 out of 23 experiments against SOURCE_λ.

Results (Q3). Table A.2 illustrates the difference between using TRADE and its single-detector variants for estimating the contamination factor γ. Overall, TRADE results in better estimates of the contamination factor (lower MAE) than the single-detector variants. In the worst case, it wins 74 times and loses 55 times compared with the IF variant.

Appendix B

Estimating the contamination factor's distribution in unsupervised anomaly detection

This Supplement contains additional details about the experiments of Chapter 5. Specifically, we provide additional details on the datasets used, and we extend the results of Q1, Q2, and Q3.

Data. Table B.1 shows the details of the used datasets. The datasets vary in terms of the number of examples (from around 50 to more than 48000), the number of covariates (from 5 to more than 1500), and the contamination factor (from around 0.004 to more than 0.10). Note that even the highest contamination factor is around 0.10, confirming the general assumption of anomalies being rare.

Q1-Q2. γGMM's estimated distribution. Table B.2 shows the MAE between γGMM's sample mean and the true value γ^* on a per-dataset basis. With respect to the true value γ^* and out of 22 experiments, the sample mean is:

- a *good estimate* (i.e., MAE ≤ 0.01) for 7 datasets (Cardiotocography, InternetAds, Pima, SpamBase, T21, WPBC, Wilt);

Dataset	N	d	γ^*	Dataset	N	d	γ^*
ALOI	12384	27	0.0304	Pima	526	8	0.0494
Annthyroid	7129	21	0.0749	Shuttle	1013	9	0.0128
Arrhythmia	271	259	0.0996	SpamBase	2661	57	0.0500
Cardiotocography	1734	21	0.0496	Stamps	340	9	0.0912
Glass	214	7	0.0421	T15	42125	10	0.0668
InternetAds	1682	1555	0.0499	T21	18509	10	0.0529
KDDCup99	48113	40	0.0042	WBC	223	9	0.0448
Lymphography	148	47	0.0405	WDBC	367	30	0.0272
PageBlocks	5473	10	0.1023	WPBC	160	33	0.0562
Parkinson	53	22	0.0943	Waveform	3443	21	0.0290
PenDigits	9868	16	0.0020	Wilt	4655	5	0.0200

Table B.1: Properties of the 22 datasets used. For each dataset, we report the number of examples, the number of original covariates, and the ground-truth contamination factor.

Dataset	γGMM's mean	γ^*	MAE	Dataset	γGMM's mean	γ^*	MAE
ALOI	0.059	0.030	0.029	Pima	0.039	0.049	0.010
Annthyroid	0.049	0.074	0.025	Shuttle	0.072	0.012	0.060
Arrhythmia	0.038	0.099	0.061	SpamBase	0.058	0.050	0.008
Cardiotocography	0.044	0.049	0.005	Stamps	0.062	0.091	0.028
Glass	0.071	0.042	0.029	T15	0.041	0.066	0.025
InternetAds	0.048	0.049	0.001	T21	0.049	0.052	0.003
KDDCup99	0.050	0.004	0.046	WBC	0.080	0.044	0.035
Lymphography	0.058	0.040	0.018	WDBC	0.065	0.027	0.038
PageBlocks	0.063	0.102	0.038	WPBC	0.063	0.056	0.006
Parkinson	0.071	0.094	0.023	Waveform	0.061	0.029	0.032
PenDigits	0.044	0.002	0.042	Wilt	0.026	0.020	0.006

Table B.2: Mean Absolute Error (MAE) between the true contamination factor and γGMM's sample mean for the 22 datasets.

- a *slightly imprecise estimate* (i.e., $0.01 < \text{MAE} \leq 0.03$) for 7 datasets (ALOI, Annthyroid, Glass, Lymphography, Parkinson, Stamps, and T15);
- a *not-optimal estimate* (i.e., $0.03 < \text{MAE} \leq 0.05$) for 6 datasets (KDDCup99, PageBlocks, PenDigits, WBC, WDBC, and Waveform);
- a *bad estimate* (MAE > 0.05) for just two datasets (Arrhythmia, and Shuttle).

This shows, again, that the estimated distribution is well-calibrated.

Q3. Selecting the anomaly detectors to compute the F_1 score. Because we aim to study the effect of the contamination factor on the detectors' performance, we

Dataset	Anomaly Detectors
ALOI	KNN
Annthyroid	HBOS
Arrhythmia	IForest-HBOS-COPOD
Cardiotocography	KNN
Glass	LOF
InternetAds	LSCP
KDDCup99	COPOD
Lymphography	KNN-LOF-OCSVM-HBOS
PageBlocks	LOF
Parkinson	LSCP-HBOS-COPOD-LSCP-HBOS-COPOD
PenDigits	KNN-IForest-LOF-OCSVM-LSCP-Ae-VAE-HBOS-LODA-COPOD
Pima	IForest
Shuttle	KNN-OCSVM-Ae-VAE-HBOS-KNN-OCSVM-Ae-VAE-HBOS
SpamBase	LSCP
Stamps	LSCP
T15	OCSVM
T21	OCSVM
WBC	KNN-LOF-OCSVM-LODA-COPOD
WDBC	KNN-LOF-OCSVM-LSCP-Ae-VAE-LODA-COPOD
WPBC	OCSVM
Waveform	OCSVM
Wilt	LOF

Table B.3: List of detectors with the greatest F_1 score when using the true contamination factor to set the threshold. For each dataset, we use such a subset of detectors to compute the deterioration.

compare the F_1 scores only over the detectors that work well for each of the datasets. For each dataset D, we use as set of detectors those achieving the greatest F_1 score using the true contamination factor, i.e. $\arg\max_{f_m} \{F_1(f_m, D, \gamma^*)\}$. This means that, for each dataset, we (1) use each detector separately to make predictions using the true contamination factor γ^*, (2) measure their F_1 score, (3) keep those detectors that obtain the greatest F_1, and (4) use them to compute the F_1 deterioration using the point-estimates of the contamination factor. Table B.3 lists the detectors used for each dataset to compute the F_1 deterioration. Observe that sometimes only a single detector obtains the greatest F_1 score, while sometimes several detectors get the same F_1 score.

Q3. False alarms and false negatives. Finally, Table B.4 shows the false alarm (false positive) rate and the false negative rate. The majority of the threshold estimators provide extremely high estimates for the contamination factor, shown here as extremely low false negative rates, but they would be useless in practice because of their high false alarm rate. In fact, this metric is important as false alarms result in real costs for the company (e.g., turning the wind turbine off to wait until the ice on the blades melts down), while reducing trust in the detection system. Our method reduces the

False Alarm Rate	
Method	Mean ± std.
IQR	0.009 ± 0.008
MTT	0.027 ± 0.024
γGMM	**0.042±0.015**
QMCD	0.059 ± 0.018
KARCH	0.147 ± 0.047
CHAU	0.190 ± 0.035
ZSCORE	0.221 ± 0.050
YJ	0.390 ± 0.139
FILTER	0.454 ± 0.054
DSN	0.477 ± 0.134
HIST	0.513 ± 0.100
FGD	0.533 ± 0.183
AUCP	0.591 ± 0.088
MCST	0.611 ± 0.287
GESD	0.616 ± 0.106
REGR	0.643 ± 0.105
MAD	0.731 ± 0.083
CLF	0.757 ± 0.077
EB	0.785 ± 0.077
WIND	0.809 ± 0.076
MOLL	0.816 ± 0.082
BOOT	0.862 ± 0.079

False Negative Rate	
Method	Mean ± std
BOOT	0.001 ± 0.002
WIND	0.001 ± 0.002
MOLL	0.001 ± 0.002
EB	0.001 ± 0.003
MAD	0.002 ± 0.003
CLF	0.002 ± 0.003
GESD	0.003 ± 0.005
REGR	0.003 ± 0.005
AUCP	0.004 ± 0.006
HIST	0.006 ± 0.008
FGD	0.007 ± 0.011
DSN	0.007 ± 0.009
FILTER	0.007 ± 0.009
MCST	0.007 ± 0.012
YJ	0.009 ± 0.010
ZSCORE	0.017 ± 0.015
CHAU	0.019 ± 0.016
KARCH	0.021 ± 0.018
QMCD	0.034 ± 0.024
γGMM	**0.036 ± 0.025**
MTT	0.042 ± 0.028
IQR	0.044 ± 0.029

Table B.4: Mean and standard deviation of the false alarm rate (left) and false negative rate (right) obtained by using each method's γ estimate to set the threshold (the lower the better). On the false alarms, γGMM has the third best mean and outperforms QMCD and KARCH, which are the second and third best baseline when measuring the F_1 score. On the other hand, γGMM obtains higher false negative rates than most competitors, because the threshold estimators overestimate the true contamination factor.

false alarm rate compared to most of the baselines, including QMCD and KARCH that achieve the second and third best F_1 scores on average. On the other hand, IQR and MTT have the lowest false positive rates, due to the fact that they often underestimate the contamination factor as supported by the false negative table.

Appendix C

Unsupervised Anomaly Detection with Rejection

In this Appendix we consider Chapter 7 and (1) provide additional theorems and proofs for Section 7.1, and (2) further describe the experimental results.

C.1 Theoretical Results

Firstly, we provide the proof for Theorem 8.

Theorem 8 (Analysis of ExCeeD) Let s be an anomaly score, and $\psi_N \in [0,1]$ the proportion of training scores $\leq s$. For $T \geq 4$, there exist $t_1 = t_1(N,\gamma,T) \in [0,1]$, $t_2 = t_2(N,\gamma,T) \in [0,1]$ such that

$$\psi_N \in [t_1, t_2] \implies \mathscr{M}_s \leq 1 - 2e^{-T}.$$

Proof. We split this proof into two parts: we show that the reverse inequalities, i.e. that **(a)** if $\psi_N \leq t_1$, then $\mathscr{M}_s \geq 1 - 2e^{-T}$, and **(b)** if $\psi_N \geq t_2$, then $\mathscr{M}_s \geq 1 - 2e^{-T}$, hold and prove the final statement because $\mathbb{P}(\hat{Y} = 1|s)$ is monotonic increasing on s.

(a) The probability $\mathbb{P}(\hat{Y} = 1|s)$ can be seen as the cumulative distribution F of a binomial random variable $\mathcal{B}(q_s, N)$ with at most $N\gamma - 1$ successes out of N trials, with $q_s = \frac{N(1-\psi_N)+1}{2+N}$ as the success probability. By applying Hoeffding's inequality, we

obtain the upper bound

$$\mathbb{P}(\hat{Y}^* = 1|s) \leq \exp\left(-2N\left(\frac{N(1-\psi_N)+1}{2+N} - \frac{N\gamma - 1}{N}\right)^2\right)$$

that holds for the constraint $\psi_N \leq \frac{2+N}{N^2} + \frac{1-2\gamma}{N} + (1-\gamma)$. Because $\mathbb{P}(\hat{Y}^* = 1|s) \leq e^{-T}$ implies that $\mathscr{M}_s \geq 1 - 2e^{-T}$, we search for the values of ψ_N such that the upper bound is $\leq e^{-T}$. Forcing the upper bound $\leq e^{-T}$ results in

$$2N\left(\frac{N(1-\psi_N)+1}{2+N} - \frac{N\gamma - 1}{N}\right)^2 - T \geq 0,$$

which is satisfied for $(I_1)\ 0 \leq \psi_N \leq A_1 - \sqrt{B_1}$ and $(I_2)\ A_1 + \sqrt{B_1} \leq \psi_N \leq 1$, where

$$A_1 = \frac{2 + N(N+1)(1-\gamma)}{N^2}, B_1 = \frac{2N\left(-3\gamma^2 - 2N(1-\gamma)^2 + 4\gamma - 3\right) + T(N+2)^2 - 8}{2N^3}.$$

However, for $T \geq 4$, no values of N, γ, and T that satisfy the constraint on ψ_N also satisfy I_2. Moving to I_1, we find out that if ψ_N satisfies I_1, then it also satisfies the constraint on ψ_N for any N, γ, and T. Therefore, we we set $t_1(N, \gamma, T) = A_1 - \sqrt{B_1}$. As a result,

$$\psi_N \leq t_1 \implies \mathbb{P}(\hat{Y}^* = 1|s) \leq e^{-T} \implies \mathscr{M}_s \geq 1 - 2e^{-T}.$$

(b) Similarly, $\mathbb{P}(\hat{Y}^* = 0|s)$ can be seen as the cumulative distribution F of $\mathcal{B}(p_s, N)$, with $N(1-\gamma)$ successes and $p_s = \frac{1+N\psi_N(s)}{2+N}$. By seeing the binomial as a sum of Bernoulli random variables, and using the property of its cumulative distribution $F(N(1-\gamma), N, p_s) + F(N\gamma - 1, N, 1 - p_s) = 1$, we apply the Hoeffding's inequality and compare such upper bound to the e^{-T}. We obtain

$$2N\left(\frac{1+\psi_N N}{2+N} - (1-\gamma)\right)^2 - T \geq 0$$

that holds with the constraint $\psi_N \geq \frac{(2+N)(1-\gamma)-1}{N}$. The quadratic inequality in ψ_N has solutions for $(I_1)\ 0 \leq \psi_N \leq A_2 - \sqrt{B_2}$ and $(I_2)\ A_2 + \sqrt{B_2} \leq \psi_N \leq 1$, where $A_2 = \frac{(2+N)(1-\gamma)-1}{N}$, and $B_2 = \frac{T(N+2)^2}{2N^3}$. However, the constraint limits the solutions to I_2, i.e. for $\psi_N \geq A_2 + \sqrt{B_2}$. Thus, we set $t_2(N, \gamma, T) = A_2 + \sqrt{B_2}$ and conclude that

$$\psi_N \geq t_2 \implies \mathbb{P}(\hat{Y}^* = 1|s) \geq 1 - e^{-T} \implies \mathscr{M}_s \geq 1 - 2e^{-T}.$$

□

Secondly, Theorem 11 relies on two important results: given S the anomaly score random variable, (1) if ψ_N was the *theoretical* cumulative of S, it would have a uniform distribution (Theorem 19), but because in practice (2) ψ_N is the *empirical* cumulative of S, its distribution is close to uniform with high probability (Theorem 20). We prove these results in the following theorems.

Theorem 19. *Let S be the anomaly score random variable, and $\psi = F_S(S)$ be the cumulative distribution of S applied to S itself. Then $\psi \sim Unif(0,1)$.*

Proof. We prove that, if $\psi \sim Unif(0,1)$, then $F_\psi(t) = t$ for any $t \in [0,1]$:

$$F_\psi(t) = \mathbb{P}(\psi \leq t) = \mathbb{P}(F_S(S) \leq t) = \mathbb{P}(S \leq F_S^{-1}(t)) = F_S(F_S^{-1}(t)) = t,$$

which implies that $\psi \sim Unif(0,1)$. □

Theorem 20. *Let ψ be as in Theorem 19, and F_{ψ_N} be its empirical distribution obtained from a sample of size N. For any small $\delta > 0$ and $t \in [0,1]$, with probability $> 1 - \delta$*

$$F_{\psi_N}(t) \in \left[F_\psi(t) - \sqrt{\frac{\ln \frac{2}{\delta}}{2N}}, F_\psi(t) + \sqrt{\frac{\ln \frac{2}{\delta}}{2N}} \right].$$

Proof. For any $\varepsilon > 0$, the DKW inequality implies

$$\mathbb{P}\left(\sup_{t \in [0,1]} |F_{\psi_N}(t) - F_\psi(t)| > \varepsilon \right) \leq 2 \exp\left(-2N\varepsilon^2\right).$$

By setting $\delta = 2 \exp\left(-2N\varepsilon^2\right)$, i.e. $\varepsilon = \sqrt{\frac{\ln \frac{2}{\delta}}{2N}}$, and using the complementary probability we conclude that

$$\mathbb{P}\left(\sup_{t \in [0,1]} |F_{\psi_N}(t) - F_\psi(t)| \leq \sqrt{\frac{\ln \frac{2}{\delta}}{2N}} \right) > 1 - \delta.$$

□

C.2 Experiments

Data. Table C.1 shows the properties of the 34 datasets used for the experimental comparison, in terms of number of examples, features, and contamination factor γ. The datasets can be downloaded in the following link: `https://github.com/Minqi824/ADBench/tree/main/datasets/Classical`.

Q1. RejEx against the baselines. Table C.2 shows the results (mean ± std) aggregated by detectors in terms of cost per example (UP) and ranking position (BOTTOM). Results confirm that RejEx obtains an average cost per example lower than all the baselines for 9 out of 12 detectors, which is similar to the runner-up SS-Repen for the remaining 3 detectors. Moreover, RejEx has always the best (lowest) average ranking position.

Dataset	N	d	γ	Dataset	N	d	γ
Aloi	20000	27	0.0315	Optdigits	5198	64	0.0254
Annthyroid	7062	6	0.0756	PageBlocks	5393	10	0.0946
Campaign	20000	62	0.1127	Pendigits	6870	16	0.0227
Cardio	1822	21	0.0960	Pima	768	8	0.3490
Cardiotocography	2110	21	0.2204	Satellite	6435	36	0.3164
Census	20000	500	0.0854	Satimage	5801	36	0.0119
Donors	20000	10	0.2146	Shuttle	20000	9	0.0725
Fault	1941	27	0.3467	Thyroid	3656	6	0.0254
Fraud	20000	29	0.0021	Vertebral	240	6	0.1250
Glass	213	7	0.0423	Vowels	1452	12	0.0317
Http	20000	3	0.0004	Waveform	3443	21	0.0290
InternetAds	1966	1555	0.1872	WBC	223	9	0.0448
Landsat	6435	36	0.2071	WDBC	367	30	0.0272
Letter	1598	32	0.0626	Wilt	4819	5	0.0533
Lymphography	148	18	0.0405	Wine	129	13	0.0775
Mammography	7848	6	0.0322	WPBC	198	33	0.2374
Musk	3062	166	0.0317	Yeast	1453	8	0.3310

Table C.1: Properties (number of examples N, features d, and contamination factor γ) of the 34 benchmark datasets used for the experiments.

Q2. Varying the costs c_{fp}, c_{fn}, c_r. Table C.3 and Table C.4 show the average cost per example and the ranking position (mean ± std) aggregated by detectors for three representative cost functions, as discussed in the paper. Results are similar in all three cases. For high false positives cost ($c_{fp} = 10$), REJEX obtains an average cost per example lower than all the baselines for 11 out of 12 detectors and always the best average ranking position. For high false negative cost ($c_{fn} = 10$) as well as for low rejection cost ($c_{fp} = 5, c_{fn} = 5, c_r = \gamma$), it has the lowest average cost for all detectors and always the best average ranking. Moreover, when rejection is highly valuable (low cost), REJEX's cost has a large gap with respect to the baselines, which means that it is particularly useful when rejection is less expensive.

Det.	**Cost per example** (mean ± std.) ×10 REJEX	SS-REPEN	MV	ENS	UDR	EM	STABILITY	NOREJECT
AE	**1.22 ± 1.39**	1.24 ± 1.33	1.36 ± 1.43	1.34 ± 1.51	1.38 ± 1.48	1.43 ± 1.50	1.43 ± 1.49	1.48 ± 1.52
COPOD	**1.23 ± 1.38**	1.25 ± 1.34	1.35 ± 1.42	1.33 ± 1.40	1.40 ± 1.44	1.43 ± 1.48	1.45 ± 1.47	1.46 ± 1.48
ECOD	**1.20 ± 1.36**	1.24 ± 1.35	1.29 ± 1.36	1.33 ± 1.43	1.39 ± 1.42	1.40 ± 1.45	1.43 ± 1.44	1.45 ± 1.45
GMM	**1.22 ± 1.35**	1.24 ± 1.37	1.44 ± 1.41	1.36 ± 1.46	1.42 ± 1.45	1.56 ± 1.48	1.51 ± 1.47	1.56 ± 1.49
HBOS	**1.16 ± 1.29**	1.23 ± 1.38	1.31 ± 1.32	1.34 ± 1.36	1.35 ± 1.37	1.39 ± 1.41	1.40 ± 1.39	1.42 ± 1.42
IF	**1.15 ± 1.28**	1.23 ± 1.35	1.30 ± 1.36	1.29 ± 1.36	1.34 ± 1.39	1.40 ± 1.43	1.39 ± 1.41	1.43 ± 1.44
INNE	**1.13 ± 1.29**	1.22 ± 1.33	1.45 ± 1.34	1.46 ± 1.40	1.45 ± 1.38	1.47 ± 1.40	1.46 ± 1.39	1.45 ± 1.40
KDE	**1.27 ± 1.40**	**1.27 ± 1.34**	1.43 ± 1.45	1.38 ± 1.45	1.44 ± 1.45	1.50 ± 1.48	1.45 ± 1.43	1.52 ± 1.48
KNNO	**1.19 ± 1.23**	1.25 ± 1.35	1.40 ± 1.31	1.35 ± 1.31	1.35 ± 1.30	1.44 ± 1.32	1.41 ± 1.31	1.46 ± 1.33
LODA	**1.25 ± 1.33**	**1.25 ± 1.34**	1.31 ± 1.30	1.39 ± 1.37	1.40 ± 1.36	1.46 ± 1.41	1.41 ± 1.31	1.51 ± 1.42
LOF	**1.26 ± 1.31**	**1.26 ± 1.36**	1.55 ± 1.40	1.40 ± 1.39	1.42 ± 1.38	1.57 ± 1.40	1.51 ± 1.39	1.58 ± 1.40
OCSVM	**1.20 ± 1.31**	1.25 ± 1.33	1.38 ± 1.38	1.32 ± 1.40	1.38 ± 1.40	1.41 ± 1.40	1.37 ± 1.36	1.47 ± 1.43
Avg.	**1.21 ± 1.33**	1.25 ± 1.35	1.38 ± 1.37	1.36 ± 1.40	1.39 ± 1.40	1.46 ± 1.43	1.44 ± 1.40	1.48 ± 1.44

Det.	**Ranking position** (mean ± std.) REJEX	SS-REPEN	MV	ENS	UDR	EM	STABILITY	NOREJECT
AE	**2.63 ± 1.63**	3.96 ± 2.94	4.78 ± 2.10	3.81 ± 2.11	4.98 ± 1.88	5.15 ± 1.80	4.92 ± 1.78	5.77 ± 1.84
COPOD	**2.49 ± 1.65**	3.44 ± 2.75	3.97 ± 1.92	4.25 ± 2.17	5.24 ± 1.70	5.13 ± 1.67	5.59 ± 1.48	5.89 ± 2.13
ECOD	**2.43 ± 1.44**	3.62 ± 2.86	3.73 ± 1.96	4.13 ± 2.29	5.53 ± 1.57	4.75 ± 1.62	5.74 ± 1.39	6.07 ± 1.93
GMM	**2.12 ± 1.08**	3.05 ± 2.49	5.26 ± 1.92	3.49 ± 2.00	4.43 ± 1.66	6.26 ± 1.24	5.20 ± 1.43	6.20 ± 1.45
HBOS	**2.29 ± 1.57**	3.64 ± 2.98	4.54 ± 2.11	4.39 ± 2.06	4.95 ± 1.79	5.04 ± 1.80	5.61 ± 1.40	5.52 ± 1.88
IF	**2.23 ± 1.48**	3.78 ± 2.78	4.12 ± 1.90	4.26 ± 2.08	5.10 ± 1.88	5.27 ± 1.66	5.34 ± 1.38	5.91 ± 2.22
INNE	**1.73 ± 1.14**	3.18 ± 2.74	5.86 ± 2.42	5.57 ± 1.40	4.94 ± 1.62	5.57 ± 1.60	5.32 ± 1.37	3.83 ± 1.63
KDE	**2.33 ± 1.42**	3.99 ± 2.86	4.74 ± 2.06	3.79 ± 2.03	5.01 ± 1.90	5.43 ± 1.59	4.87 ± 1.92	5.84 ± 1.80
KNNO	**2.02 ± 1.29**	3.58 ± 2.87	4.87 ± 1.81	3.94 ± 1.94	4.41 ± 1.83	5.75 ± 1.49	5.22 ± 1.62	6.21 ± 1.48
LODA	**2.89 ± 1.77**	3.17 ± 2.30	4.15 ± 2.26	4.36 ± 2.14	4.99 ± 2.00	5.50 ± 2.04	4.98 ± 2.11	5.95 ± 1.73
LOF	**2.04 ± 1.01**	3.16 ± 2.73	5.68 ± 1.40	3.32 ± 1.71	3.96 ± 1.63	6.15 ± 1.19	5.47 ± 1.49	6.22 ± 1.31
OCSVM	**2.33 ± 1.29**	3.92 ± 2.84	4.89 ± 1.98	3.85 ± 2.17	4.86 ± 1.89	5.31 ± 1.80	5.06 ± 1.89	5.78 ± 1.66
Avg.	**2.29 ± 1.40**	3.54 ± 2.76	4.72 ± 1.99	4.10 ± 2.01	4.87 ± 1.78	5.44 ± 1.63	5.28 ± 1.60	5.77 ± 1.76

Table C.2: ***Top*:** Cost per example (mean ± std) ×10 per detector aggregated over the datasets. Results show that REJEX obtains a lower average cost for 9 out of 12 detectors and a similar average cost as the runner-up SS-REPEN for the remaining 3 detectors. Moreover, REJEX has the best overall average (last row). ***Bottom*:** Ranking positions (mean ± std) per detector aggregated over the datasets. Results show that REJEX obtains always the lowest average rank, despite being close to the runner-up SS-REPEN when the detector is LODA.

Det.	REJEX	SS-REPEN	Cost per Example for Three Cost Functions (mean ± std) MV	ENS	UBR	EM	STABILITY	NOREJECT
False Positive Cost = 10, False Negative Cost = 1, Rejection Cost = $\min\{10(1-\gamma), \gamma\}$								
AE	**0.504 ± 0.626**	0.584 ± 0.723	0.697 ± 0.763	0.661 ± 0.830	0.703 ± 0.829	0.766 ± 0.841	0.768 ± 0.826	0.825 ± 0.873
COPOD	**0.491 ± 0.637**	0.593 ± 0.706	0.686 ± 0.746	0.618 ± 0.726	0.707 ± 0.788	0.778 ± 0.825	0.785 ± 0.801	0.781 ± 0.833
ECOD	**0.479 ± 0.628**	0.584 ± 0.727	0.625 ± 0.705	0.642 ± 0.755	0.711 ± 0.774	0.748 ± 0.803	0.770 ± 0.783	0.771 ± 0.817
GMM	**0.568 ± 0.713**	0.589 ± 0.752	0.823 ± 0.878	0.715 ± 0.929	0.790 ± 0.925	0.941 ± 0.948	0.889 ± 0.929	0.950 ± 0.967
HBOS	**0.475 ± 0.595**	0.569 ± 0.758	0.666 ± 0.693	0.697 ± 0.732	0.709 ± 0.764	0.776 ± 0.803	0.771 ± 0.770	0.809 ± 0.816
IF	**0.477 ± 0.602**	0.575 ± 0.712	0.665 ± 0.718	0.634 ± 0.731	0.683 ± 0.786	0.776 ± 0.818	0.763 ± 0.788	0.808 ± 0.831
INNE	**0.479 ± 0.592**	0.567 ± 0.698	0.752 ± 0.724	0.820 ± 0.795	0.815 ± 0.787	0.819 ± 0.793	0.818 ± 0.792	0.823 ± 0.799
KDE	0.602 ± 0.827	**0.589 ± 0.704**	0.819 ± 0.947	0.740 ± 0.913	0.793 ± 0.939	0.897 ± 0.945	0.774 ± 0.906	0.914 ± 0.945
KNNO	**0.498 ± 0.577**	0.596 ± 0.726	0.741 ± 0.734	0.669 ± 0.720	0.669 ± 0.736	0.777 ± 0.747	0.739 ± 0.735	0.800 ± 0.749
LODA	**0.518 ± 0.619**	0.574 ± 0.709	0.574 ± 0.647	0.689 ± 0.729	0.701 ± 0.748	0.762 ± 0.774	0.697 ± 0.682	0.827 ± 0.797
LOF	**0.539 ± 0.623**	0.603 ± 0.742	0.898 ± 0.840	0.685 ± 0.773	0.715 ± 0.790	0.891 ± 0.813	0.831 ± 0.821	0.887 ± 0.808
OCSVM	**0.479 ± 0.599**	0.589 ± 0.705	0.745 ± 0.790	0.632 ± 0.752	0.694 ± 0.782	0.760 ± 0.775	0.695 ± 0.737	0.818 ± 0.806
False Positive Cost = 1, False Negative Cost = 10, Rejection Cost = $\min\{1-\gamma, 10\gamma\}$								
AE	**0.730 ± 0.747**	0.761 ± 0.756	0.909 ± 0.882	0.784 ± 0.825	0.780 ± 0.805	0.819 ± 0.843	0.789 ± 0.825	0.797 ± 0.821
COPOD	**0.761 ± 0.767**	0.765 ± 0.770	0.930 ± 0.888	0.794 ± 0.805	0.800 ± 0.801	0.844 ± 0.842	0.802 ± 0.815	0.827 ± 0.832
ECOD	**0.739 ± 0.759**	0.767 ± 0.766	0.900 ± 0.858	0.789 ± 0.811	0.788 ± 0.787	0.840 ± 0.839	0.791 ± 0.803	0.821 ± 0.819
GMM	**0.670 ± 0.676**	0.765 ± 0.767	0.845 ± 0.782	0.754 ± 0.755	0.739 ± 0.736	0.785 ± 0.757	0.760 ± 0.753	0.766 ± 0.750
HBOS	**0.687 ± 0.684**	0.776 ± 0.782	0.824 ± 0.808	0.750 ± 0.768	0.744 ± 0.747	0.785 ± 0.787	0.749 ± 0.765	0.753 ± 0.766
IF	**0.679 ± 0.680**	0.775 ± 0.776	0.847 ± 0.824	0.755 ± 0.771	0.743 ± 0.745	0.761 ± 0.772	0.757 ± 0.774	0.763 ± 0.770
INNE	**0.660 ± 0.685**	0.772 ± 0.779	0.695 ± 0.620	0.774 ± 0.742	0.748 ± 0.722	0.758 ± 0.737	0.744 ± 0.716	0.773 ± 0.754
KDE	**0.691 ± 0.692**	0.791 ± 0.773	0.887 ± 0.836	0.754 ± 0.760	0.755 ± 0.744	0.785 ± 0.807	0.758 ± 0.754	0.759 ± 0.760
KNNO	**0.706 ± 0.657**	0.767 ± 0.764	0.839 ± 0.779	0.791 ± 0.736	0.769 ± 0.710	0.778 ± 0.736	0.799 ± 0.736	0.803 ± 0.729
LODA	**0.750 ± 0.714**	0.781 ± 0.775	0.880 ± 0.850	0.811 ± 0.768	0.806 ± 0.761	0.804 ± 0.783	0.827 ± 0.780	0.838 ± 0.784
LOF	**0.738 ± 0.679**	0.764 ± 0.764	0.999 ± 0.833	0.826 ± 0.757	0.810 ± 0.739	0.871 ± 0.770	0.867 ± 0.799	0.846 ± 0.747
OCSVM	**0.730 ± 0.711**	0.780 ± 0.774	0.953 ± 0.878	0.791 ± 0.786	0.795 ± 0.773	0.845 ± 0.833	0.787 ± 0.772	0.796 ± 0.783
False Positive Cost = 5, False Negative Cost = 5, Rejection Cost = γ								
AE	**0.534 ± 0.611**	0.618 ± 0.666	0.671 ± 0.716	0.644 ± 0.741	0.655 ± 0.736	0.707 ± 0.748	0.705 ± 0.740	0.738 ± 0.762
COPOD	**0.545 ± 0.619**	0.627 ± 0.673	0.676 ± 0.724	0.629 ± 0.674	0.666 ± 0.716	0.719 ± 0.747	0.719 ± 0.730	0.731 ± 0.739
ECOD	**0.529 ± 0.609**	0.625 ± 0.675	0.629 ± 0.687	0.638 ± 0.702	0.662 ± 0.705	0.701 ± 0.736	0.708 ± 0.716	0.724 ± 0.727
GMM	**0.534 ± 0.599**	0.626 ± 0.687	0.719 ± 0.709	0.656 ± 0.716	0.675 ± 0.720	0.776 ± 0.736	0.746 ± 0.731	0.780 ± 0.743
HBOS	**0.499 ± 0.572**	0.622 ± 0.694	0.632 ± 0.661	0.650 ± 0.669	0.641 ± 0.681	0.695 ± 0.706	0.688 ± 0.686	0.710 ± 0.709
IF	**0.497 ± 0.569**	0.623 ± 0.677	0.643 ± 0.680	0.620 ± 0.667	0.629 ± 0.692	0.696 ± 0.712	0.688 ± 0.701	0.714 ± 0.719
INNE	**0.491 ± 0.569**	0.617 ± 0.668	0.622 ± 0.572	0.718 ± 0.691	0.691 ± 0.674	0.709 ± 0.685	0.705 ± 0.675	0.726 ± 0.698
KDE	**0.562 ± 0.639**	0.642 ± 0.673	0.709 ± 0.739	0.666 ± 0.711	0.684 ± 0.726	0.748 ± 0.742	0.679 ± 0.689	0.761 ± 0.742
KNNO	**0.521 ± 0.544**	0.628 ± 0.677	0.687 ± 0.667	0.657 ± 0.646	0.634 ± 0.651	0.701 ± 0.664	0.693 ± 0.657	0.728 ± 0.664
LODA	**0.550 ± 0.595**	0.627 ± 0.677	0.601 ± 0.649	0.670 ± 0.668	0.665 ± 0.680	0.708 ± 0.698	0.683 ± 0.645	0.757 ± 0.711
LOF	**0.554 ± 0.580**	0.628 ± 0.681	0.809 ± 0.737	0.678 ± 0.682	0.674 ± 0.688	0.792 ± 0.703	0.759 ± 0.718	0.788 ± 0.698
OCSVM	**0.523 ± 0.582**	0.631 ± 0.671	0.704 ± 0.728	0.634 ± 0.685	0.657 ± 0.695	0.709 ± 0.704	0.660 ± 0.674	0.733 ± 0.716

Table C.3: Cost per example (mean ± std) per detector aggregated over the datasets. The table is divided into three parts, where each part has different costs (false positives, false negatives, rejection). Results show that REJEX obtains a lower average cost in all cases but one (KDE).

Det.	REJEX	SS-REPEN	MV	ENS	UDR	EM	STABILITY	NOREJECT
	Rankings for the Three Cost Functions (mean ± std)							
	False Positive Cost = 10, **False Negative Cost** = 1, **Rejection Cost** = $\min\{10(1-\gamma), \gamma\}$							
AE	**2.35 ± 1.37**	3.84 ± 2.73	5.87 ± 2.40	3.68 ± 2.05	4.85 ± 1.96	5.33 ± 1.67	4.72 ± 1.68	5.36 ± 1.97
COPOD	**2.25 ± 1.45**	3.63 ± 2.66	4.79 ± 2.27	3.89 ± 2.27	4.94 ± 1.76	5.46 ± 1.71	5.51 ± 1.45	5.54 ± 2.17
ECOD	**2.28 ± 1.30**	3.51 ± 2.73	4.63 ± 2.36	3.85 ± 2.21	5.34 ± 1.74	5.11 ± 1.66	5.38 ± 1.39	5.90 ± 2.09
GMM	**2.13 ± 0.99**	3.10 ± 2.43	6.36 ± 2.23	3.31 ± 1.94	4.21 ± 1.69	6.28 ± 1.27	5.18 ± 1.53	5.44 ± 1.50
HBOS	**2.12 ± 1.42**	3.46 ± 2.85	5.41 ± 2.42	4.25 ± 2.06	4.78 ± 1.78	5.35 ± 1.66	5.42 ± 1.47	5.20 ± 1.95
IF	**2.11 ± 1.49**	3.69 ± 2.61	4.73 ± 2.26	4.02 ± 2.11	5.07 ± 1.87	5.39 ± 1.61	5.36 ± 1.41	5.63 ± 2.24
INNE	**1.72 ± 1.24**	3.09 ± 2.68	5.42 ± 2.45	6.16 ± 1.44	5.47 ± 1.59	4.60 ± 1.51	5.32 ± 1.31	4.21 ± 1.80
KDE	**2.14 ± 1.25**	3.82 ± 2.68	5.73 ± 2.36	3.54 ± 1.92	4.75 ± 1.85	5.84 ± 1.58	4.83 ± 1.91	5.36 ± 1.75
KNNO	**1.99 ± 1.28**	3.50 ± 2.74	5.55 ± 2.21	3.92 ± 2.02	4.37 ± 1.86	5.73 ± 1.46	5.23 ± 1.68	5.71 ± 1.71
LODA	**2.56 ± 1.53**	3.31 ± 2.31	4.29 ± 2.48	4.34 ± 2.14	5.03 ± 1.95	5.42 ± 1.88	4.96 ± 2.04	6.08 ± 1.75
LOF	**1.96 ± 1.03**	3.04 ± 2.46	7.12 ± 1.28	3.14 ± 1.60	3.73 ± 1.31	6.27 ± 1.14	5.59 ± 1.66	5.15 ± 1.39
OCSVM	**2.15 ± 1.20**	3.93 ± 2.70	5.92 ± 2.30	3.58 ± 2.13	4.70 ± 1.92	5.40 ± 1.63	5.03 ± 1.89	5.29 ± 1.67
	False Positive Cost = 1, **False Negative Cost** = 10, **Rejection Cost** = $\min\{1-\gamma, 10\gamma\}$							
AE	**2.98 ± 1.93**	3.82 ± 2.72	7.03 ± 1.95	4.30 ± 2.07	4.49 ± 1.82	4.96 ± 1.83	4.28 ± 1.62	4.14 ± 1.93
COPOD	**2.91 ± 2.04**	3.56 ± 2.69	7.13 ± 1.56	4.15 ± 1.97	4.43 ± 1.87	5.30 ± 1.86	4.50 ± 1.60	4.03 ± 1.79
ECOD	**2.70 ± 1.96**	3.88 ± 2.82	6.87 ± 1.72	4.15 ± 2.02	4.74 ± 1.79	5.01 ± 2.03	4.23 ± 1.50	4.42 ± 1.90
GMM	**2.59 ± 1.70**	3.99 ± 2.85	6.84 ± 2.22	4.04 ± 2.12	4.08 ± 1.77	5.73 ± 1.37	4.62 ± 1.55	4.12 ± 1.58
HBOS	**2.96 ± 2.14**	4.32 ± 2.93	6.41 ± 2.20	4.49 ± 1.92	4.37 ± 1.82	5.15 ± 1.94	4.48 ± 1.65	3.81 ± 1.81
IF	**2.71 ± 2.06**	4.51 ± 2.92	6.80 ± 2.00	4.47 ± 2.09	4.62 ± 1.72	4.33 ± 1.66	4.55 ± 1.47	4.01 ± 1.93
INNE	**2.64 ± 1.94**	4.71 ± 2.93	5.06 ± 2.95	5.85 ± 1.52	5.06 ± 1.80	4.03 ± 1.64	4.72 ± 1.50	3.94 ± 1.87
KDE	**3.00 ± 2.01**	4.49 ± 2.93	6.51 ± 2.27	4.00 ± 1.84	4.40 ± 1.68	5.01 ± 1.97	4.40 ± 1.78	4.18 ± 1.96
KNNO	**2.64 ± 2.01**	4.11 ± 3.01	6.67 ± 2.23	4.17 ± 1.89	4.13 ± 1.87	4.99 ± 1.60	4.88 ± 1.54	4.41 ± 1.64
LODA	**3.44 ± 1.96**	3.66 ± 2.71	6.32 ± 2.30	4.22 ± 1.94	4.36 ± 1.95	4.47 ± 2.17	4.53 ± 2.09	4.99 ± 1.87
LOF	**2.22 ± 1.38**	3.43 ± 2.67	7.74 ± 0.67	3.47 ± 1.73	3.63 ± 1.40	5.95 ± 1.17	5.35 ± 1.57	4.22 ± 1.38
OCSVM	**2.82 ± 1.71**	3.83 ± 2.63	7.30 ± 1.50	4.23 ± 2.13	4.35 ± 1.78	5.34 ± 1.65	4.32 ± 1.95	3.80 ± 1.72
	False Positive Cost = 5, **False Negative Cost** = 5, **Rejection Cost** = γ							
AE	**2.31 ± 1.38**	4.05 ± 2.78	5.85 ± 2.41	3.66 ± 2.12	4.69 ± 1.86	5.27 ± 1.68	4.84 ± 1.74	5.34 ± 1.92
COPOD	**2.24 ± 1.49**	3.72 ± 2.70	4.62 ± 2.17	3.98 ± 2.34	4.89 ± 1.87	5.26 ± 1.58	5.57 ± 1.50	5.72 ± 2.10
ECOD	**2.18 ± 1.31**	3.92 ± 2.75	4.22 ± 2.22	3.93 ± 2.33	5.30 ± 1.82	4.91 ± 1.69	5.48 ± 1.38	6.06 ± 1.94
GMM	**1.96 ± 0.97**	3.31 ± 2.44	6.39 ± 2.21	3.36 ± 1.89	4.09 ± 1.68	6.29 ± 1.25	5.11 ± 1.51	5.48 ± 1.50
HBOS	**1.98 ± 1.37**	3.95 ± 2.88	5.30 ± 2.46	4.18 ± 2.01	4.63 ± 1.83	5.36 ± 1.68	5.41 ± 1.46	5.18 ± 1.93
IF	**2.01 ± 1.46**	4.10 ± 2.67	4.71 ± 2.26	3.96 ± 2.05	4.98 ± 1.96	5.35 ± 1.60	5.29 ± 1.42	5.60 ± 2.24
INNE	**1.70 ± 1.25**	3.75 ± 2.82	4.17 ± 2.42	6.02 ± 1.44	5.57 ± 1.78	4.56 ± 1.36	5.36 ± 1.38	4.87 ± 2.08
KDE	**2.22 ± 1.35**	4.24 ± 2.73	5.49 ± 2.55	3.62 ± 2.00	4.71 ± 1.95	5.60 ± 1.50	4.79 ± 1.86	5.34 ± 1.87
KNNO	**1.98 ± 1.23**	3.88 ± 2.82	5.49 ± 2.39	3.91 ± 1.86	4.29 ± 1.86	5.56 ± 1.71	5.19 ± 1.63	5.69 ± 1.70
LODA	**2.58 ± 1.60**	3.59 ± 2.36	4.34 ± 2.58	4.26 ± 2.17	4.93 ± 1.98	5.33 ± 1.98	5.02 ± 1.98	5.94 ± 1.75
LOF	**1.88 ± 0.96**	3.26 ± 2.51	7.18 ± 1.29	3.16 ± 1.60	3.65 ± 1.32	6.24 ± 1.12	5.53 ± 1.69	5.10 ± 1.37
OCSVM	**2.15 ± 1.19**	4.18 ± 2.77	5.73 ± 2.36	3.62 ± 2.20	4.59 ± 1.88	5.39 ± 1.59	5.05 ± 1.92	5.31 ± 1.67

Table C.4: Rankings (mean ± std) per detector aggregated over the datasets, where lower positions mean lower costs (better). The table is divided into three parts, where each part has different costs for false positives, false negatives, and rejection. REJEX obtains the lowest (best) average ranking position for all the detectors and all cost functions.

Bibliography

[1] M. R. Abbas, M. S. A. Nadeem, A. Shaheen, A. A. Alshdadi, R. Alharbey, S.-O. Shim, and W. Aziz. "Accuracy rejection normalized-cost curves (ARNCCs): A novel 3-dimensional framework for robust classification". In: *IEEE Access* 7 (2019), pp. 160125–160143.

[2] N. Abe, B. Zadrozny, and J. Langford. "Outlier detection by active learning". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2006, pp. 504–509.

[3] B. Afsari. "Riemannian L^p center of mass: existence, uniqueness, and convexity". In: *Proceedings of the American Mathematical Society* 139.2 (2011), pp. 655–673.

[4] C. C. Aggarwal. "An introduction to outlier analysis". In: *Outlier analysis*. Springer, 2017, pp. 1–34.

[5] M. Ahmed, N. Choudhury, and S. Uddin. "Anomaly detection on big data in financial markets". In: *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. 2017, pp. 998–1001.

[6] M. Ahmed, A. N. Mahmood, and M. R. Islam. "A survey of anomaly detection techniques in financial domain". In: *Future Generation Computer Systems* 55 (2016), pp. 278–288.

[7] R. Alaiz-Rodríguez and N. Japkowicz. "Assessing the impact of changing environments on classifier performance". In: *Advances in Artificial Intelligence: 21st Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2008 Windsor, Canada, May 28-30, 2008 Proceedings 21*. Springer. 2008, pp. 13–24.

[8] M. J. Alrawashdeh. "An adjusted Grubbs' and Generalized Extreme Studentized Deviation". In: *Demonstratio Mathematica* 54.1 (2021), pp. 548–557.

[9] D. Amagata, M. Onizuka, and T. Hara. "Fast and exact outlier detection in metric spaces: a proximity graph-based approach". In: *Proceedings of the 2021 International Conference on Management of Data*. 2021, pp. 36–48.

[10] F. Angiulli and C. Pizzuti. "Fast outlier detection in high dimensional spaces". In: *European conference on principles of data mining and knowledge discovery*. Springer. 2002, pp. 15–27.

[11] N. Archana and S. Pawar. "Periodicity Detection of Outlier Sequences using Constraint Based Pattern Tree with MAD". In: *International Journal of Advanced Studies in Computers, Science and Engineering* 4.6 (2015), p. 34.

[12] V. Bagdonavičius and L. Petkevičius. "Multiple outlier detection tests for parametric models". In: *Mathematics* 8.12 (2020), p. 2156.

[13] T. R. Bandaragoda, K. M. Ting, D. Albrecht, F. T. Liu, Y. Zhu, and J. R. Wells. "Isolation-based anomaly detection using nearest-neighbor ensembles". In: *Computational Intelligence* 34.4 (2018), pp. 968–998.

[14] S. Bansod and A. Nandedkar. “Transfer learning for video anomaly detection”. In: *Journal of Intelligent & Fuzzy Systems* 36.3 (2019), pp. 1967–1975.

[15] J.-M. Bardet and S.-F. Dimby. “A new non-parametric detector of univariate outliers for distributions with unbounded support”. In: *Extremes* 20.4 (2017), pp. 751–775.

[16] R. E. Barlow and H. D. Brunk. “The isotonic regression problem and its dual”. In: *Journal of the American Statistical Association* 67.337 (1972), pp. 140–147.

[17] V. Barnabé-Lortie, C. Bellinger, and N. Japkowicz. “Active learning for one-class classification”. In: *14th International Conference on Machine Learning and Applications*. IEEE. 2015, pp. 390–395.

[18] C. Baur, B. Wiestler, M. Muehlau, C. Zimmer, N. Navab, and S. Albarqouni. “Modeling healthy anatomy with artificial intelligence for unsupervised anomaly detection in brain MRI”. In: *Radiology: Artificial Intelligence* 3.3 (2021).

[19] J. Bekker and J. Davis. “Estimating the class prior in positive and unlabeled data through decision tree induction”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018.

[20] J. Bekker and J. Davis. “Learning from positive and unlabeled data under the selected at random assumption”. In: *Second International Workshop on Learning with Imbalanced Domains: Theory and Applications*. PMLR. 2018, pp. 8–22.

[21] J. Bekker and J. Davis. “Learning from positive and unlabeled data: A survey”. In: *Machine Learning* (2020).

[22] J. Bekker, P. Robberechts, and J. Davis. “Beyond the Selected Completely At Random Assumption for Learning from Positive and Unlabeled Data”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2019.

[23] A. Bella, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. “Calibration of machine learning models”. In: *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI Global, 2010, pp. 128–146.

[24] D. I. Belov and R. D. Armstrong. “Distributions of the Kullback–Leibler divergence with applications”. In: *British Journal of Mathematical and Statistical Psychology* 64.2 (2011), pp. 291–309.

[25] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger. “MVTec AD–A comprehensive real-world dataset for unsupervised anomaly detection”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 9592–9600.

[26] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. “Variational Inference: A review for statisticians”. In: *Journal of the American statistical Association* 112.518 (2017), pp. 859–877.

[27] L. Bol’shev and M. Ubaidullaeva. “Chauvenet’s test in the classical theory of errors”. In: *Theory of Probability & Its Applications* 19.4 (1975), pp. 683–692.

[28] R. J. Bolton, D. J. Hand, et al. “Unsupervised profiling methods for fraud detection”. In: *Credit scoring and credit control VII* (2001), pp. 235–255.

[29] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. “LOF: identifying density-based local outliers”. In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, pp. 93–104.

[30] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

[31] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. “Understanding disentangling in β-VAE”. In: *arXiv preprint arXiv:1804.03599* (2018).

[32] P.-C. Bürkner and M. Vuorre. “Ordinal regression models in psychology: A tutorial”. In: *Advances in Methods and Practices in Psychological Science* 2.1 (2019), pp. 77–101.

[33] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle. “On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study”. In: *Data Mining and Knowledge Discovery* 30.4 (2016), pp. 891–927.

[34] V. Cerqueira, L. Torgo, and C. Soares. “Early anomaly detection in time series: a hierarchical approach for predicting critical health episodes”. In: *Machine Learning* (2023), pp. 1–22.

[35] R. Chalapathy and S. Chawla. “Deep learning for anomaly detection: A survey”. In: *arXiv preprint arXiv:1901.03407* (2019).

[36] V. Chandola, A. Banerjee, and V. Kumar. “Anomaly detection: A survey”. In: *ACM computing surveys (CSUR)* 41.3 (2009), p. 15.

[37] N. Charoenphakdee, Z. Cui, Y. Zhang, and M. Sugiyama. “Classification with rejection based on cost-sensitive classification”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 1507–1517.

[38] H. Chen, F. Liu, Y. Wang, L. Zhao, and H. Wu. “A variational approach for learning from positive and unlabeled data”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 14844–14854.

[39] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau. “Autoencoder-based network Anomaly Detection”. In: *2018 Wireless telecommunications symposium (WTS)*. IEEE. 2018, pp. 1–5.

[40] C. Chow. “On optimum recognition error and reject tradeoff”. In: *IEEE Transactions on information theory* 16.1 (1970), pp. 41–46.

[41] C.-K. Chow. “An optimum character recognition system using decision functions”. In: *IRE Transactions on Electronic Computers* 4 (1957), pp. 247–254.

[42] M. Christoffel, G. Niu, and M. Sugiyama. “Class-prior estimation for learning from positive and unlabeled data”. In: *Asian Conference on Machine Learning*. PMLR. 2016, pp. 221–236.

[43] D. Coin. “Testing normality in the presence of outliers”. In: *Statistical Methods and Applications* 17.1 (2008), pp. 3–12.

[44] D. Conte, P. Foggia, G. Percannella, A. Saggese, and M. Vento. “An ensemble of rejecting classifiers for anomaly detection of audio events”. In: *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance*. IEEE. 2012, pp. 76–81.

[45] A. A. Cook, G. Mısırlı, and Z. Fan. “Anomaly detection for IoT time-series data: A survey”. In: *IEEE Internet of Things Journal* 7.7 (2019), pp. 6481–6494.

[46] C. Corbiere, N. Thome, A. Saporta, T.-H. Vu, M. Cord, and P. Perez. “Confidence estimation via auxiliary models”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.10 (2021), pp. 6043–6055.

[47] L. P. Cordella, C. De Stefano, C. Sansone, and M. Vento. “An adaptive reject option for LVQ classifiers”. In: *Image Analysis and Processing: 8th International Conference, ICIAP’95 San Remo, Italy, September 13–15, 1995 Proceedings 8*. Springer. 1995, pp. 68–73.

[48] C. Cortes, G. DeSalvo, C. Gentile, M. Mohri, and S. Yang. “Online learning with abstention”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1059–1067.

[49] C. Cortes, G. DeSalvo, and M. Mohri. “Learning with rejection”. In: *International Conference on Algorithmic Learning Theory*. Springer. 2016.

[50] J. Davis, E. S. Burnside, I. de Castro Dutra, D. Page, R. Ramakrishnan, V. S. Costa, and J. W. Shavlik. “View Learning for Statistical Relational Learning: With an Application to Mammography.” In: *IJCAI*. Citeseer. 2005, pp. 677–683.

[51] J. Demšar. “Statistical comparisons of classifiers over multiple datasets”. In: *Journal of Machine learning research* 7.Jan (2006), pp. 1–30.

[52] J. Deng and E. T. Brown. “SSDBCODI: Semi-Supervised Density-Based Clustering with Outliers Detection Integrated”. In: *arXiv preprint arXiv:2208.05561* (2022).

[53] C. Denis and M. Hebiri. “Consistency of plug-in confidence sets for classification in semi-supervised learning”. In: *Journal of Nonparametric Statistics* 32.1 (2020), pp. 42–72.

[54] L. Devos, L. Perini, W. Meert, and J. Davis. “Detecting Evasion Attacks in Deployed Tree Ensembles”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2023, pp. 120–136.

[55] B. Du, L. Zhang, D. Tao, and D. Zhang. “Unsupervised transfer learning for target detection from hyperspectral images”. In: *Neurocomputing* 120 (2013), pp. 72–82.

[56] M. C. Du Plessis, G. Niu, and M. Sugiyama. “Analysis of learning from positive and unlabeled data”. In: *Advances in neural information processing systems* 27 (2014).

[57] S. Duan, L. Matthey, A. Saraiva, N. Watters, C. P. Burgess, A. Lerchner, and I. Higgins. “Unsupervised model selection for variational disentangled representation learning”. In: *arXiv preprint arXiv:1905.12614* (2019).

[58] B. Dubuisson and M. Masson. “A statistical decision rule with incomplete knowledge about classes”. In: *Pattern recognition* 26.1 (1993), pp. 155–165.

[59] O. J. Dunn. “Multiple comparisons among means”. In: *Journal of the American statistical association* 56.293 (1961), pp. 52–64.

[60] D. B. Dunson and J.-H. Park. “Kernel Stick-Breaking processes”. In: *Biometrika* 95.2 (2008), pp. 307–323.

[61] I. El Naqa. “Detection and prediction of radiotherapy errors”. In: *Machine Learning in Radiation Oncology: Theory and Applications* (2015), pp. 237–241.

[62] C. Elkan and K. Noto. “Learning classifiers from only positive and unlabeled data”. In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008, pp. 213–220.

[63] A. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong. “A meta-analysis of the Anomaly Detection problem”. In: *arXiv preprint arXiv:1503.01158* (2015).

[64] A. Farahani, S. Voghoei, K. Rasheed, and H. R. Arabnia. “A brief review of domain adaptation”. In: *Advances in data science and information engineering: proceedings from ICDATA 2020 and IKE 2020* (2021), pp. 877–894.

[65] T. S. Ferguson. “A Bayesian analysis of some nonparametric problems”. In: *The annals of statistics* (1973), pp. 209–230.

[66] D. Fink. “A compendium of conjugate priors”. In: *See http://www. people. cornell. edu/pages/df36/CONJINTRnew% 20TEX. pdf* 46 (1997).

[67] H. Fischer. *A history of the central limit theorem: from classical to modern probability theory*. Vol. 4. Springer, 2011.

[68] P. Flach and M. Kull. “Precision-recall-gain curves: PR analysis done right”. In: *Advances in neural information processing systems*. Vol. 28. 2015.

[69] R. Foorthuis. “On the nature and types of anomalies: a review of deviations in data”. In: *International journal of data science and analytics* 12.4 (2021), pp. 297–331.

[70] D. Fourure, M. U. Javaid, N. Posocco, and S. Tihon. “Anomaly detection: How to artificially increase your f1-score with a biased evaluation protocol”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2021, pp. 3–18.

[71] P. I. Frazier. “A tutorial on Bayesian optimization”. In: *arXiv preprint arXiv:1807.02811* (2018).

[72] M. Friendly, G. Monette, and J. Fox. “Elliptical insights: understanding statistical methods through elliptical geometry”. In: *Statistical Science* 28.1 (2013), pp. 1–39.

[73] F. Gao, J. Li, R. Cheng, Y. Zhou, and Y. Ye. “Connet: Deep semi-supervised anomaly detection based on sparse positive samples”. In: *IEEE Access* 9 (2021), pp. 67249–67258.

[74] J. Gao and P.-N. Tan. “Converting output scores from outlier detection algorithms into probability estimates”. In: *International Conference on Data Mining*. IEEE. 2006, pp. 212–221.

[75] A. Garrido. “About some properties of the Kullback-Leibler divergence”. In: *Advanced Modeling and Optimization* 11.4 (2009).

[76] Y. Geifman and R. El-Yaniv. “Selectivenet: A deep neural network with an integrated reject option”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2151–2159.

[77] M.-I. Georgescu, A. Barbalau, R. T. Ionescu, F. S. Khan, M. Popescu, and M. Shah. “Anomaly detection in video via self-supervised and multi-task learning”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 12742–12752.

[78] W. Gerych, T. Hartvigsen, L. Buquicchio, E. Agu, and E. Rundensteiner. “Recovering the propensity score from biased positive unlabeled data”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 6. 2022, pp. 6694–6702.

[79] A. Ghasemi, H. R. Rabiee, M. Fadaee, M. T. Manzuri, and M. H. Rohban. “Active learning from positive and unlabeled data”. In: *2011 IEEE 11th International Conference on Data Mining Workshops*. IEEE. 2011, pp. 244–250.

[80] A. L. Gibbs and F. E. Su. “On choosing and bounding probability metrics”. In: *International statistical review* 70.3 (2002), pp. 419–435.

[81] N. Goix. “How to evaluate the quality of unsupervised anomaly detection algorithms?” In: *arXiv preprint arXiv:1607.01152* (2016).

[82] M. Goldstein and A. Dengel. “Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm”. In: *KI-2012: Poster and Demo Track* (2012), pp. 59–63.

[83] M. Goldstein and S. Uchida. “A comparative evaluation of unsupervised Anomaly Detection algorithms for multivariate data”. In: *PloS one* 11.4 (2016), e0152173.

[84] K. Golmohammadi and O. R. Zaiane. “Time series contextual anomaly detection for detecting market manipulation in stock market”. In: *2015 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE. 2015, pp. 1–10.

[85] A. Goodge, B. Hooi, S.-K. Ng, and W. S. Ng. “Lunar: Unifying local outlier detection methods via graph neural networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 2022, pp. 6737–6745.

[86] N. Görnitz, M. Kloft, K. Rieck, and U. Brefeld. “Toward supervised anomaly detection”. In: *Journal of Artificial Intelligence Research* 46 (2013), pp. 235–262.

[87] D. Görür and E. C. Rasmussen. “Dirichlet Process Gaussian Mixture Models: Choice of the base distribution”. In: *Journal of Computer Science and Technology* 25.4 (2010), pp. 653–664.

[88] C. Goutte and E. Gaussier. “A probabilistic interpretation of precision, recall and F-score, with implication for evaluation”. In: *Advances in Information Retrieval: 27th European Conference on IR Research, ECIR 2005, Santiago de Compostela, Spain, March 21-23, 2005. Proceedings 27*. Springer. 2005, pp. 345–359.

[89] P. J. Green and S. Richardson. “Modelling heterogeneity with and without the Dirichlet Process”. In: *Scandinavian journal of statistics* 28.2 (2001), pp. 355–375.

[90] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. “On calibration of modern neural networks”. In: *International Conference on Machine Learning*. 2017, pp. 1321–1330.

[91] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao. “Adbench: Anomaly detection benchmark”. In: *Advances in Neural Information Processing Systems*. Vol. 35. 2022, pp. 32142–32159.

[92] X. Han, X. Chen, and L.-P. Liu. “Gan ensemble for anomaly detection”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 5. 2021, pp. 4090–4097.

[93] B. Hanczar. “Performance visualization spaces for classification with rejection option”. In: *Pattern Recognition* 96 (2019), p. 106984.

[94] J. A. Hanley and B. J. McNeil. “A method of comparing the areas under receiver operating characteristic curves derived from the same cases.” In: *Radiology* 148.3 (1983), pp. 839–843.

[95] J. A. Hanley and B. J. McNeil. “The meaning and use of the area under a receiver operating characteristic (ROC) curve.” In: *Radiology* 143.1 (1982), pp. 29–36.

[96] S. A. Haque, M. Rahman, and S. M. Aziz. “Sensor anomaly detection in wireless sensor networks for healthcare”. In: *Sensors* 15.4 (2015), pp. 8764–8786.

[97] F. Harrou, Y. Sun, F. Kadri, S. Chaabane, and C. Tahon. “Early detection of abnormal patient arrivals at hospital emergency department”. In: *2015 International Conference on Industrial Engineering and Systems Management (IESM)*. IEEE. 2015, pp. 221–227.

[98] N. Hashemi, E. V. German, J. P. Ramirez, and J. Ruths. “Filtering approaches for dealing with noise in Anomaly Detection”. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 5356–5361.

[99] F. He, T. Liu, G. I. Webb, and D. Tao. “Instance-dependent pu learning by bayesian optimal relabeling”. In: *arXiv preprint arXiv:1808.02180* (2018).

[100] G. He, Y. Duan, Y. Li, T. Qian, J. He, and X. Jia. “Active learning for multivariate time series classification with positive unlabeled data”. In: *Proceedings of the 27th International Conference on Tools with Artificial Intelligence*. 2015, pp. 178–185.

[101] Z. He, X. Xu, and S. Deng. “Discovering cluster-based local outliers”. In: *Pattern Recognition Letters* 24.9-10 (2003), pp. 1641–1650.

[102] N. A. Heard, D. J. Weston, K. Platanioti, and D. J. Hand. “Bayesian Anomaly Detection methods for social networks”. In: *The Annals of Applied Statistics* 4 (2010).

[103] M. E. Hellman. “The nearest neighbor classification rule with a reject option”. In: *IEEE Transactions on Systems Science and Cybernetics* 6.3 (1970), pp. 179–185.

[104] K. Hendrickx, W. Meert, B. Cornelis, and J. Davis. “Know your limits: Machine learning with rejection for vehicle engineering”. In: *International Conference on Advanced Data Mining and Applications*. Springer. 2022, pp. 273–288.

[105] K. Hendrickx, L. Perini, D. Van der Plas, W. Meert, and J. Davis. “Machine learning with a reject option: A survey”. In: *Machine Learning* (2024).

[106] H. Hoffmann. “Kernel PCA for novelty detection”. In: *Pattern recognition* 40.3 (2007), pp. 863–874.

[107] H. Hojjati, T. K. K. Ho, and N. Armanfard. “Self-Supervised Anomaly Detection: A Survey and Outlook”. In: *arXiv preprint arXiv:2205.05173* (2022).

[108] Y. Hou, R. He, J. Dong, Y. Yang, and W. Ma. “IoT Anomaly Detection Based on Autoencoder and Bayesian Gaussian Mixture Model”. In: *Electronics* 11.20 (2022), p. 3287.

[109] Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira. “Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10951–10960.

[110] L. Huang, C. Zhang, and H. Zhang. “Self-adaptive training: beyond empirical risk minimization”. In: *Advances in neural information processing systems*. Vol. 33. 2020, pp. 19365–19376.

[111] T. Idé, D. T. Phan, and J. Kalagnanam. “Multi-task multi-modal models for collective anomaly detection”. In: *International Conference on Data Mining*. IEEE. 2017, pp. 177–186.

[112] R. L. Iman and J. M. Davenport. “Approximations of the critical region of the friedman statistic”. In: *Communications in Statistics-Theory and Methods* 9.6 (1980), pp. 571–595.

[113] D. Iouchtchenko, N. Raymond, P.-N. Roy, and M. Nooijen. “Deterministic and quasi-random sampling of optimized Gaussian Mixture distributions for Vibronic Monte Carlo”. In: *arXiv preprint arXiv:1912.11594* (2019).

[114] A. Jacobson, L. Kavan, and O. Sorkine-Hornung. “Robust inside-outside segmentation using generalized winding numbers”. In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), pp. 1–12.

[115] N. Japkowicz, C. Myers, M. Gluck, et al. “A novelty detection approach to classification”. In: *IJCAI*. Vol. 1. Citeseer. 1995, pp. 518–523.

[116] M. Jiang, C. Hou, A. Zheng, X. Hu, S. Han, H. Huang, X. He, P. S. Yu, and Y. Zhao. “Weakly supervised anomaly detection: A survey”. In: *arXiv preprint arXiv:2302.04549* (2023).

[117] X. Jin, Y. Guo, S. Sarkar, A. Ray, and R. M. Edwards. “Anomaly detection in nuclear power plants via symbolic dynamic filtering”. In: *IEEE Transactions on Nuclear Science* 58.1 (2010), pp. 277–288.

[118] H. Jmila and M. I. Khedher. “Adversarial machine learning for network intrusion detection: A comparative study”. In: *Computer Networks* (2022), p. 109073.

[119] M. I. Jordan and T. M. Mitchell. “Machine learning: Trends, perspectives, and prospects”. In: *Science* 349.6245 (2015), pp. 255–260.

[120] I. A. Karatepe and E. Zeydan. “Anomaly detection in cellular network data using big data analytics”. In: *European Wireless 2014; 20th European Wireless Conference*. VDE. 2014, pp. 1–5.

[121] M. Kato, T. Teshima, and J. Honda. “Learning from positive and unlabeled data with a selection bias”. In: *International conference on learning representations*. 2018.

[122] M. A. Keyzer and B. Sonneveld. “Using the mollifier method to characterize datasets and models: the case of the Universal Soil Loss Equation”. In: *ITC Journal* 3.4 (1997), pp. 263–272.

[123] D. P. Kingma and M. Welling. “Auto-encoding Variational Bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[124] M. A. Kocak, D. Ramirez, E. Erkip, and D. E. Shasha. “Safepredict: A meta-algorithm for machine learning that uses refusals to guarantee correctness”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.2 (2019), pp. 663–678.

[125] S. Kok and P. Domingos. “Learning the structure of Markov logic networks”. In: *International Conference on Machine Learning*. 2005, pp. 441–448.

[126] Ł. Korycki, A. Cano, and B. Krawczyk. “Active learning with abstaining classifiers for imbalanced drifting data streams”. In: *2019 IEEE international conference on big data (big data)*. IEEE. 2019, pp. 2334–2343.

[127] W. M. Kouw and M. Loog. “A review of domain adaptation without target labels”. In: *IEEE transactions on pattern analysis and machine intelligence* 43.3 (2019), pp. 766–785.

[128] W. M. Kouw and M. Loog. “An introduction to domain adaptation and transfer learning”. In: *arXiv preprint arXiv:1812.11806* (2018).

[129] H.-P. Kriegel, P. Kroger, E. Schubert, and A. Zimek. “Interpreting and unifying outlier scores”. In: *International Conference on Data Mining*. 2011, pp. 13–24.

[130] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. “Outlier detection in axis-parallel subspaces of high dimensional data”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2009, pp. 831–838.

[131] M. Kubat, R. C. Holte, and S. Matwin. “Machine learning for the detection of oil spills in satellite radar images”. In: *Machine learning* 30 (1998), pp. 195–215.

[132] M. Kull and P. Flach. “Novel decompositions of proper scoring rules for classification: Score adjustment as precursor to calibration”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2015, pp. 68–85.

[133] M. Kull, M. P. Nieto, M. Kängsepp, T. Silva Filho, H. Song, and P. Flach. “Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with Dirichlet calibration”. In: *Advances in Neural Information Processing Systems*. 2019.

[134] M. Kull, T. Silva Filho, and P. Flach. “Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers”. In: *Artificial Intelligence and Statistics*. 2017, pp. 623–631.

[135] M. Kull, T. M. Silva Filho, P. Flach, et al. “Beyond sigmoids: How to obtain well-calibrated probabilities from binary classifiers with beta calibration”. In: *Electronic Journal of Statistics* 11.2 (2017), pp. 5052–5080.

[136] A. Kumagai, T. Iwata, and Y. Fujiwara. “Transfer anomaly detection by inferring latent domain representations”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 2471–2481.

[137] R. Kumari and S. K. Srivastava. “Machine learning: A review on binary classification”. In: *International Journal of Computer Applications* 160.7 (2017).

[138] T. C. Landgrebe, D. M. Tax, P. Paclik, R. P. Duin, and C. Andrew. “A combining strategy for ill-defined problems”. In: *Fifteenth Annual Symposium of the Pattern Recognition Association of South Africa*. 2004, pp. 57–62.

[139] S. Laroui, X. Descombes, A. Vernay, F. Villiers, F. Villalba, and E. Debreuve. “How to define a rejection class based on model learning?” In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 569–576.

[140] L. J. Latecki, A. Lazarevic, and D. Pokrajac. “Outlier detection with kernel density functions”. In: *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer. 2007, pp. 61–75.

[141] C.-L. Li, K. Sohn, J. Yoon, and T. Pfister. “Cutpaste: Self-supervised learning for anomaly detection and localization”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 9664–9674.

[142] S. Li, X. Ji, E. Dobriban, O. Sokolsky, and I. Lee. “PAC-Wrap: Semi-Supervised PAC Anomaly Detection”. In: *arXiv preprint arXiv:2205.10798* (2022).

[143] W. Li, G. Wu, and Q. Du. “Transferred deep learning for anomaly detection in hyperspectral imagery”. In: *IEEE Geoscience and Remote Sensing Letters* 14.5 (2017), pp. 597–601.

[144] Z. Li, J. Li, Y. Wang, and K. Wang. “A deep learning approach for anomaly detection based on SAE and LSTM in mechanical equipment”. In: *The International Journal of Advanced Manufacturing Technology* 103 (2019), pp. 499–510.

[145] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu. “COPOD: copula-based outlier detection”. In: *International Conference on Data Mining*. IEEE. 2020.

[146] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, and G. Chen. “Ecod: Unsupervised outlier detection using empirical cumulative distribution functions”. In: *IEEE Transactions on Knowledge and Data Engineering* (2022).

[147] J. Lin. “On the Dirichlet distribution”. In: *Department of Mathematics and Statistics, Queens University* (2016).

[148] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. “Focal loss for dense object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[149] C. Lioma, J. G. Simonsen, and B. Larsen. “Evaluation measures for relevance and credibility in ranked lists”. In: *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*. 2017, pp. 91–98.

[150] F. T. Liu, K. M. Ting, and Z.-H. Zhou. “Isolation forest”. In: *International Conference on Data Mining*. IEEE. 2008, pp. 413–422.

[151] F. T. Liu, K. M. Ting, and Z.-H. Zhou. “Isolation-based Anomaly Detection”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6.1 (2012), pp. 1–39.

[152] K. Liu, M. Zhu, H. Fu, H. Ma, and T.-S. Chua. “Enhancing Anomaly Detection in Surveillance Videos with Transfer Learning from Action Recognition”. In: *Proceedings of the 28th ACM International Conference on Multimedia*. 2020, pp. 4664–4668.

[153] M. Q. Ma, Y. Zhao, X. Zhang, and L. Akoglu. “A large-scale study on unsupervised outlier model selection: Do internal strategies suffice?” In: *arXiv preprint arXiv:2104.01422* (2021).

[154] H. O. Marques, R. J. Campello, J. Sander, and A. Zimek. “Internal evaluation of unsupervised outlier detection”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14.4 (2020), pp. 1–42.

[155] H. O. Marques, R. J. Campello, A. Zimek, and J. Sander. “On the internal evaluation of unsupervised outlier detection”. In: *Proceedings of the 27th international conference on scientific and statistical database management*. 2015, pp. 1–12.

[156] C. Marrocco, M. Molinara, and F. Tortorella. “An empirical comparison of ideal and empirical ROC-based reject rules”. In: *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer. 2007, pp. 47–60.

[157] P.-F. Marteau, S. Soheily-Khah, and N. Béchet. “Hybrid isolation forest-application to intrusion detection”. In: *arXiv preprint arXiv:1705.03800* (2017).

[158] T. Martens, L. Perini, and J. Davis. “Semi-supervised Learning from Active Noisy Soft Labels for Anomaly Detection”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2023, pp. 219–236.

[159] L. Martí, N. Sanchez-Pi, J. M. Molina, and A. C. B. Garcia. “Anomaly Detection based on sensor data in petroleum industry applications”. In: *Sensors* 15.2 (2015), pp. 2774–2797.

[160] M. A. Martin and S. Roberts. “An evaluation of bootstrap methods for outlier detection in least squares regression”. In: *Journal of Applied Statistics* 33.7 (2006), pp. 703–720.

[161] R. A. Maxion and R. R. Roberts. “Proper use of ROC curves in Intrusion/Anomaly Detection”. In: *School of Computing Science Technical Report Series* (2004).

[162] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici. “N-baiot—network-based detection of iot botnet attacks using deep autoencoders”. In: *IEEE Pervasive Computing* 17.3 (2018), pp. 12–22.

[163] A. K. Menon and R. C. Williamson. “A loss framework for calibrated anomaly detection”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 1494–1504.

[164] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai. “Kitsune: an ensemble of autoencoders for online network intrusion detection”. In: *arXiv preprint arXiv:1802.09089* (2018).

[165] I. Monroy, G. Escudero, and M. Graells. “Anomaly detection in batch chemical processes”. In: *Computer Aided Chemical Engineering*. Vol. 26. Elsevier, 2009, pp. 255–260.

[166] M. P. Naeini, G. Cooper, and M. Hauskrecht. “Obtaining well calibrated probabilities using Bayesian binning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2015.

[167] M. P. Naeini and G. F. Cooper. “Binary classifier calibration using an ensemble of near isotonic regression models”. In: *International Conference on Data Mining*. IEEE. 2016, pp. 360–369.

[168] R. M. Neal. “Bayesian Mixture Modeling”. In: *Maximum Entropy and Bayesian Methods*. Springer, 1992, pp. 197–211.

[169] V.-L. Nguyen and E. Hullermeier. “Reliable multilabel classification: Prediction with partial abstention”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 2020, pp. 5264–5271.

[170] K. Nian, H. Zhang, A. Tayal, T. Coleman, and Y. Li. "Auto insurance fraud detection using unsupervised spectral ranking for anomaly". In: *The Journal of Finance and Data Science* 2.1 (2016), pp. 58–75.

[171] A. Niculescu-Mizil and R. Caruana. "Obtaining Calibrated Probabilities from Boosting." In: *UAI*. Vol. 5. 2005, pp. 413–20.

[172] A. Niculescu-Mizil and R. Caruana. "Predicting good probabilities with supervised learning". In: *International Conference on Machine Learning*. 2005, pp. 625–632.

[173] S. Niu, Y. Liu, J. Wang, and H. Song. "A decade survey of transfer learning (2010–2020)". In: *IEEE Transactions on Artificial Intelligence* 1.2 (2020), pp. 151–166.

[174] S. W. Nydick. "The Wishart and inverse Wishart distributions". In: *Electronic Journal of Statistics* 6.1-19 (2012).

[175] N. Otsu. "A threshold selection method from gray-level histograms". In: *IEEE transactions on systems, man, and cybernetics* 9.1 (1979), pp. 62–66.

[176] B. Ozenne, F. Subtil, and D. Maucort-Boulch. "The precision–recall curve overcame the optimism of the receiver operating characteristic curve in rare diseases". In: *Journal of clinical epidemiology* 68.8 (2015), pp. 855–859.

[177] G. Pang, L. Cao, L. Chen, and H. Liu. "Learning representations of ultrahigh-dimensional data for random distance-based outlier detection". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 2041–2050.

[178] G. Pang, C. Shen, H. Jin, and A. van den Hengel. "Deep weakly-supervised anomaly detection". In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2023, pp. 1795–1807.

[179] S. Papadopoulos, A. Drosou, and D. Tzovaras. "A novel graph-based descriptor for the detection of billing-related anomalies in cellular mobile networks". In: *IEEE Transactions on Mobile Computing* 15.11 (2016), pp. 2655–2668.

[180] M. Perello-Nieto, E. S. Telmo De Menezes Filho, M. Kull, and P. Flach. "Background Check: A general technique to build more reliable and versatile classifiers". In: *International Conference on Data Mining*. IEEE. 2016.

[181] L. Perini, P.-C. Bürkner, and A. Klami. "Estimating the contamination factor's distribution in unsupervised anomaly detection". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 27668–27679.

[182] L. Perini and J. Davis. "Unsupervised Anomaly Detection with Rejection". In: *Advances in Neural Information Processing Systems*. 2023.

[183] L. Perini, C. Galvin, and V. Vercruyssen. "A Ranking Stability Measure for Quantifying the Robustness of Anomaly Detection Methods". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2020, pp. 397–408.

[184] L. Perini, D. Giannuzzi, and J. Davis. "How to Allocate your Label Budget? Choosing between Active Learning and Learning to Reject in Anomaly Detection". In: *arXiv preprint arXiv:2301.02909* (2023).

[185] L. Perini, V. Vercruyssen, and J. Davis. "Class Prior Estimation in Active Positive and Unlabeled Learning." In: *Proceedings of the 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence (IJCAI-PRICAI)*. 2020.

[186] L. Perini, V. Vercruyssen, and J. Davis. "Learning from Positive and Unlabeled Multi-Instance Bags in Anomaly Detection". In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2023, pp. 1897–1906.

[187] L. Perini, V. Vercruyssen, and J. Davis. “Quantifying the confidence of anomaly detectors in their example-wise predictions”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2020.

[188] L. Perini, V. Vercruyssen, and J. Davis. “Transferring the Contamination Factor between Anomaly Detection Domains by Shape Similarity”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 2022, pp. 4128–4136.

[189] T. Pevnỳ. “Loda: Lightweight on-line detector of anomalies”. In: *Machine Learning* 102.2 (2016), pp. 275–304.

[190] J. Platt et al. “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods”. In: *Advances in large margin classifiers* (1999).

[191] M. Prastawa, E. Bullitt, S. Ho, and G. Gerig. “A brain tumor segmentation framework based on outlier detection”. In: *Medical image analysis* 8.3 (2004), pp. 275–283.

[192] A. Pugnana and S. Ruggieri. “AUC-based Selective Classification”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2023, pp. 2494–2514.

[193] Z. Qi, D. Jiang, and X. Chen. “Iterative gradient descent for outlier detection”. In: *International Journal of Wavelets, Multiresolution and Information Processing* 19.04 (2021), p. 2150004.

[194] H. Ramaswamy, C. Scott, and A. Tewari. “Mixture proportion estimation via kernel embeddings of distributions”. In: *International Conference on Machine Learning*. 2016, pp. 2052–2060.

[195] S. Ramaswamy, R. Rastogi, and K. Shim. “Efficient algorithms for mining outliers from large datasets”. In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, pp. 427–438.

[196] D. Ramotsoela, A. Abu-Mahfouz, and G. Hancke. “A survey of anomaly detection in industrial wireless sensor networks with critical water system infrastructure as a case study”. In: *Sensors* 18.8 (2018), p. 2491.

[197] R. B. Randall. *Vibration-based condition monitoring: industrial, aerospace and automotive applications*. John Wiley & Sons, 2011.

[198] C. Rasmussen. “The infinite Gaussian Mixture Model”. In: *Advances in neural information processing systems*. Vol. 12. 1999.

[199] S. Rayana and L. Akoglu. “Less is more: Building selective anomaly ensembles”. In: *Acm transactions on knowledge discovery from data (tkdd)* 10.4 (2016), pp. 1–33.

[200] J. Raymaekers and P. J. Rousseeuw. “Transforming variables to central normality”. In: *Machine Learning* (2021), pp. 1–23.

[201] K. Ren, H. Yang, Y. Zhao, W. Chen, M. Xue, H. Miao, S. Huang, and J. Liu. “A robust AUC maximization framework with simultaneous outlier detection and feature selection for positive-unlabeled classification”. In: *IEEE transactions on neural networks and learning systems* 30.10 (2018), pp. 3072–3083.

[202] D. Rengasamy, B. C. Rothwell, and G. P. Figueredo. “Towards a more reliable interpretation of machine learning outputs for safety-critical systems using feature importance fusion”. In: *Applied Sciences* 11.24 (2021), p. 11854.

[203] M. Rezapour. “Anomaly detection using unsupervised methods: credit card fraud case study”. In: *International Journal of Advanced Computer Science and Applications* 10.11 (2019).

[204] E. Roberts, B. A. Bassett, and M. Lochner. “Bayesian Anomaly Detection and Classification”. In: *arXiv preprint arXiv:1902.08627* (2019).

[205] S. J. Roberts, D. Husmeier, I. Rezek, and W. Penny. “Bayesian approaches to Gaussian Mixture Modeling”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.11 (1998), pp. 1133–1142.

[206] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft. “Deep semi-supervised anomaly detection”. In: *arXiv preprint arXiv:1906.02694* (2019).

[207] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. “Estimating the support of a high-dimensional distribution”. In: *Neural computation* (2001).

[208] E. Schubert, R. Wojdanowski, A. Zimek, and H.-P. Kriegel. “On evaluation of outlier rankings and outlier scores”. In: *International Conference on Data Mining*. SIAM. 2012, pp. 1047–1058.

[209] C. Scott and G. Blanchard. *Transductive Anomaly Detection*. Tech. rep. Tech. Rep., 2008, http://www. eecs. umich. edu/cscott, 2008.

[210] V. Sehwag, M. Chiang, and P. Mittal. “Ssd: A unified framework for self-supervised outlier detection”. In: *arXiv preprint arXiv:2103.12051* (2021).

[211] B. Settles. “Active learning”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6.1 (2012), pp. 1–114.

[212] B. Settles. *Active learning literature survey*. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences, 2009.

[213] S. Shekhar, M. Ghavamzadeh, and T. Javidi. “Binary classification with bounded abstention rate”. In: *arXiv preprint arXiv:1905.09561* (2019).

[214] Y. Shen and G. Cooper. “A new prior for Bayesian Anomaly Detection”. In: *Methods of Information in Medicine* 49.01 (2010), pp. 44–53.

[215] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang. *A novel anomaly detection scheme based on principal component classifier*. Tech. rep. Miami Univ Coral Gables Fl Dept of Electrical and Computer Engineering, 2003.

[216] T. Silva Filho, H. Song, M. Perello-Nieto, R. Santos-Rodriguez, M. Kull, and P. Flach. “Classifier calibration: a survey on how to assess and improve predicted class probabilities”. In: *Machine Learning* (2023), pp. 1–50.

[217] K. Singh and S. Upadhyaya. “Outlier detection: applications and techniques”. In: *International Journal of Computer Science Issues (IJCSI)* 9.1 (2012), p. 307.

[218] T. Sipes, S. Jiang, K. Moore, N. Li, H. Karimabadi, and J. R. Barr. “Anomaly detection in healthcare: Detecting erroneous treatment plans in time series radiotherapy data”. In: *International Journal of Semantic Computing* 8.03 (2014), pp. 257–278.

[219] A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans. “Probabilities for sv machines”. In: (2000).

[220] J. Soenen, E. Van Wolputte, L. Perini, V. Vercruyssen, W. Meert, J. Davis, and H. Blockeel. “The Effect of Hyperparameter Tuning on the Comparative Evaluation of Unsupervised Anomaly Detection Methods”. In: *Proceedings of the KDD’21 Workshop on Outlier Detection and Description*. Outlier Detection and Description Organising Committee. 2021, pp. 1–9.

[221] A. P. Soms. “Exact confidence intervals, based on the Z statistic, for the difference between two proportions”. In: *Communications in Statistics-Simulation and Computation* 18.4 (1989), pp. 1325–1341.

[222] R. Storn and K. Price. “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces”. In: *Journal of global optimization* 11.4 (1997), pp. 341–359.

[223] L. Stradiotti, L. Perini, and J. Davis. “Semi-Supervised Isolation Forest for Anomaly Detection”. In: *International Conference on Data Mining*. SIAM. 2024.

[224] M. Sugiyama, S. Nakajima, H. Kashima, P. Buenau, and M. Kawanabe. “Direct importance estimation with model selection and its application to covariate shift adaptation”. In: *Advances in neural information processing systems*. Vol. 20. 2007.

[225] B. Sun, J. Feng, and K. Saenko. “Correlation alignment for unsupervised domain adaptation”. In: *Domain Adaptation in Computer Vision Applications*. Springer, 2017, pp. 153–171.

[226] K. Thanammal, R. Vijayalakshmi, S. Arumugaperumal, and J. Jayasudha. “Effective Histogram Thresholding Techniques for Natural Images Using Segmentation”. In: *Journal of Image and Graphics* 2.2 (2014), pp. 113–116.

[227] N. Toron, J. Mourão-Miranda, and J. Shawe-Taylor. “TransductGAN: a Transductive Adversarial Model for Novelty Detection”. In: *arXiv e-prints* (2022), arXiv–2203.

[228] H. Trittenbach, A. Englhardt, and K. Böhm. “An overview and a benchmark of active learning for outlier detection with One-Class classifiers”. In: *arXiv preprint arXiv:1808.04759* (2018).

[229] H. Trittenbach, A. Englhardt, and K. Böhm. “Validating One-Class Active Learning with User Studies–a Prototype and Open Challenges”. In: *ECML PKDD Workshop*. 2019, p. 17.

[230] A. Ukil, S. Bandyoapdhyay, C. Puri, and A. Pal. “IoT healthcare analytics: The importance of anomaly detection”. In: *2016 IEEE 30th international conference on advanced information networking and applications (AINA)*. IEEE. 2016, pp. 994–997.

[231] J. Vaicenavicius, D. Widmann, C. Andersson, F. Lindsten, J. Roll, and T. B. Schön. “Evaluating model calibration in classification”. In: *arXiv:1902.06977* (2019).

[232] D. Van der Plas, W. Meert, J. Verbraecken, and J. Davis. “A reject option for automated sleep stage scoring”. In: *Workshop on Interpretable ML in Healthcare at International Conference on Machine Learning (ICML)*. 2021.

[233] S. Vargaftik, I. Keslassy, A. Orda, and Y. Ben-Itzhak. “RADE: resource-efficient supervised anomaly detection using decision tree-based ensemble methods”. In: *Machine Learning* 110.10 (2021), pp. 2835–2866.

[234] V. Vercruyssen, W. Meert, and J. Davis. “Transfer learning for anomaly detection through localized and unsupervised instance selection”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 2020, pp. 6054–6061.

[235] V. Vercruyssen, W. Meert, and J. Davis. “Transfer learning for time series anomaly detection”. In: *CEUR Workshop Proceedings*. Vol. 1924. 2017, pp. 27–37.

[236] V. Vercruyssen, L. Perini, W. Meert, and J. Davis. “Multi-domain Active Learning for Semi-supervised Anomaly Detection”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2022, pp. 485–501.

[237] V. Vercruyssen, M. Wannes, V. Gust, M. Koen, B. Ruben, and D. Jesse. “Semi-supervised Anomaly Detection with an Application to Water Analytics”. In: *International Conference on Data Mining*. 2018.

[238] A. Verma, A. Taneja, and A. Arora. “Fraud detection and frequent pattern matching in insurance claims using data mining techniques”. In: *2017 tenth international conference on contemporary computing (IC3)*. IEEE. 2017, pp. 1–7.

[239] M. E. Villa-Pérez, M. A. Alvarez-Carmona, O. Loyola-Gonzalez, M. A. Medina-Pérez, J. C. Velazco-Rossell, and K.-K. R. Choo. “Semi-supervised anomaly detection algorithms: A comparative summary and future research directions”. In: *Knowledge-Based Systems* 218 (2021), p. 106878.

[240] W. Wang, Q. Chen, X. He, and L. Tang. “Cooperative Anomaly Detection With Transfer Learning-Based Hidden Markov Model in Virtualized Network Slicing”. In: *IEEE Communications Letters* 23.9 (2019), pp. 1534–1537.

[241] X. Wang and S.-M. Yiu. “Classification with Rejection: Scaling Generative Classifiers with Supervised Deep Infomax.” In: *IJCAI*. 2020, pp. 2980–2986.

[242] K. Weiss, T. M. Khoshgoftaar, and D. Wang. “A survey of transfer learning”. In: *Journal of Big data* 3.1 (2016), pp. 1–40.

[243] J. Wen, C.-N. Yu, and R. Greiner. “Robust learning under uncertain test distributions: Relating covariate shift to model misspecification”. In: *International Conference on Machine Learning*. PMLR. 2014, pp. 631–639.

[244] T. Wen and R. Keyes. “Time series anomaly detection using convolutional neural networks and transfer learning”. In: *arXiv preprint arXiv:1905.13628* (2019).

[245] L. Xiang, X. Yang, A. Hu, H. Su, and P. Wang. “Condition monitoring and anomaly detection of wind turbine based on cascaded and bidirectional deep learning networks”. In: *Applied Energy* 305 (2022), p. 117925.

[246] P. Xiong, Y. Zhu, Z. Sun, Z. Cao, M. Wang, Y. Zheng, J. Hou, T. Huang, and Z. Que. “Application of transfer learning in continuous time series for anomaly detection in commercial aircraft flight data”. In: *2018 IEEE International Conference on Smart Cloud*. IEEE. 2018, pp. 13–18.

[247] H. Xu, G. Pang, Y. Wang, and Y. Wang. “Deep isolation forest for anomaly detection”. In: *IEEE Transactions on Knowledge and Data Engineering* (2023).

[248] M. Yamaguchi, Y. Koizumi, and N. Harada. “AdaFlow: Domain-adaptive density estimator with application to anomaly detection and unpaired cross-domain translation”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 3647–3651.

[249] R. El-Yaniv et al. “On the Foundations of Noise-free Selective Classification.” In: *Journal of Machine Learning Research* 11.5 (2010).

[250] B. Zadrozny and C. Elkan. “Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers”. In: *International Conference on Machine Learning*. 2001, pp. 609–616.

[251] B. Zadrozny and C. Elkan. “Transforming classifier scores into accurate multiclass probability estimates”. In: *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2002, pp. 694–699.

[252] A. Zaher, S. McArthur, D. Infield, and Y. Patel. “Online wind turbine fault detection through automated SCADA data analysis”. In: *Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology* 12.6 (2009), pp. 574–593.

[253] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang. “Deep structured energy based models for anomaly detection”. In: *International conference on machine learning*. PMLR. 2016, pp. 1100–1109.

[254] L. Zhang, K. Liu, Y. Wang, and Z. B. Omariba. “Ice detection model of wind turbine blades based on random forest classifier”. In: *Energies* 11.10 (2018), p. 2548.

[255] H. Zhao, H. Liu, W. Hu, and X. Yan. “Anomaly detection and fault analysis of wind turbine components based on deep learning network”. In: *Renewable energy* 127 (2018), pp. 825–834.

[256] N. Zhao, X. Wen, and S. Li. “A review on gas turbine anomaly detection for implementing health management”. In: *Turbo Expo: Power for Land, Sea, and Air* (2016).

[257] Y. Zhao, Z. Nasrullah, M. K. Hryniewicki, and Z. Li. “LSCP: Locally selective combination in parallel outlier ensembles”. In: *International Conference on Data Mining*. SIAM. 2019, pp. 585–593.

[258] Y. Zhao, Z. Nasrullah, and Z. Li. “PyOD: A Python Toolbox for Scalable Outlier Detection”. In: *Journal of Machine Learning Research* 20 (2019), pp. 1–7.

[259] Y. Zhou, X. Song, Y. Zhang, F. Liu, C. Zhu, and L. Liu. “Feature encoding with autoencoders for weakly supervised anomaly detection”. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.6 (2021), pp. 2454–2465.

[260] Z.-H. Zhou. *Machine learning*. Springer Nature, 2021.

[261] Y. Zhu, Z. Dong, Z. Cheng, X. Huang, Y. Dong, and Z. Zhang. “Neural network extended state-observer for energy system monitoring”. In: *Energy* 263 (2023), p. 125736.

[262] A. Zimek, R. J. Campello, and J. Sander. “Ensembles for unsupervised outlier detection: challenges and research questions a position paper”. In: *ACM SIGKDD Explorations Newsletter* 15.1 (2014), pp. 11–22.

[263] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen. “Deep Autoencoding Gaussian Mixture Model for unsupervised Anomaly Detection”. In: *International conference on learning representations*. 2018.

Curriculum Vitae

Lorenzo Perini was born in Florence, Italy, on July 2nd, 1995. In July 2017, he earned a Bachelor's degree in Mathematics from the Università degli Studi di Firenze with a thesis on Stochastic Differential Equations. In July 2019, he obtained a Master's degree in Mathematical Engineering from Politecnico di Torino, specializing in Statistics on Data and Network Optimization. Collaborating with the local company Tierra S.p.A., he conducted his Master's thesis on Predictive Maintenance for off-road vehicles, utilizing Hidden Markov Models and Autoencoders for trend Anomaly Detection.

He started his Ph.D. under the supervision of Prof. Dr. Jesse Davis in the Machine Learning group within the Declarative Languages and Artificial Intelligence (DTAI) lab. After two years, he received a Ph.D. fellowship for fundamental research from the Research Foundation – Flanders (FWO) and the Scientific Prize Gustave Boël-Sofina Fellowship for talented researchers for a long stay abroad. Thanks to the latter, he served as a visiting researcher at the University of Helsinki (Finland), joining the Multi-Source Probabilistic Inference (MUPI) research group led by Prof. Dr. Arto Klami. During that period, he also worked as a teaching assistant during the Nordic Probabilistic Summer School. Towards the end of his Ph.D., he completed a four-month industrial internship at Bosch GmbH in Stuttgart (Germany) where he held the position of Machine Learning Research Scientist. Finally, on 28/03/2024, he successfully defended his doctoral dissertation entitled *Operational, Uncertainty-Aware, and Reliable Anomaly Detection*.

List of publications

Conference Papers

Perini, L., Vercruyssen, V. and Davis, J. Class prior estimation in active positive and unlabeled learning. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence (IJCAI-PRICAI 2020)*.

Perini, L., Vercruyssen, V. and Davis, J. Transferring the Contamination Factor between Anomaly Detection Domains by Shape Similarity. In *Proceedings of the Thirty-Six AAAI Conference on Artificial Intelligence (AAAI 2022)*.

Perini, L., Bürkner, P.-C. and Klami, A. Estimating the contamination factor's distribution in unsupervised anomaly detection. In *Proceedings of the Fortieth International Conference on Machine Learning (ICML 2023)*.

Perini, L., Vercruyssen, V. and Davis, J. Quantifying the confidence of anomaly detectors in their example-wise predictions. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2020)*.

Perini, L. and Davis, J. Unsupervised Anomaly Detection with Rejection. In *Proceedings of the Thirty-Seven Conference on Neural Information Processing Systems (NeurIPS 2023)*.

Perini, L., Vercruyssen, V. and Davis, J. Learning from Positive and Unlabeled Multi-Instance Bags in Anomaly Detection. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD 2023)*.

Stradiotti, L., Perini, L. and Davis, J. Semi-Supervised Isolation Forest for Anomaly Detection. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM 2024)*.

Devos, L., Perini, L., Meert, W. and Davis, J. Detecting Evasion Attacks in Deployed

Tree Ensembles. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2023)*.

Martens, T., Perini, L. and Davis, J. Semi-supervised Learning from Active Noisy Soft Labels for Anomaly Detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2023)*.

Vercruyssen, V., Perini, L., Meert, W. and Davis, J. Multi-domain Active Learning for Semi-supervised Anomaly Detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2022)*.

Journal Papers

Hendrickx, K.[1], Perini, L.[1], Van der Plas, D., Meert, W. and Davis, J. Machine learning with a reject option: A survey. *Machine Learning*, 2024.

Papers Under Review

Perini, L., Rudolph, M., Schmedding, S. and Qiu, C. Uncertainty-aware Evaluation of Auxiliary Anomalies with the Expected Anomaly Posterior. Submitted to *International Conference on Machine Learning (ICML 2024)*.

Pugnana, A., Perini, L., Davis, J. and Ruggieri, S. Deep Neural Network Benchmarks for Selective Classification. Submitted to the *Journal of Data-centric Machine Learning Research (DMLR)*.

Workshop Papers

Soenen, J., Van Wolputte, E., Perini, L., Vercruyssen, V., Meert, W., Davis, J. and Blockeel, H.. The effect of hyperparameter tuning on the comparative evaluation of unsupervised anomaly detection methods. In *Proceedings of the KDD'21 Workshop on Outlier Detection and Description (OOD 2021 Workshop at SIGKDD 2021)*.

Perini, L., Galvin, C. and Vercruyssen, V. A Ranking Stability Measure for Quantifying the Robustness of Anomaly Detection Methods. In *Proceedings of the 2nd Workshop on Evaluation and Experimental Design in Data Mining and Machine Learning (EDML 2020 Workshop at ECML-PKDD 2020)*.

Perini, L., Giannuzzi, D. and Davis, J. How to Allocate your Label Budget? Choosing between Active Learning and Learning to Reject in Anomaly Detection. In *1st AAAI Workshop on Uncertainty Reasoning and Quantification in Decision Making*. arXiv preprint arXiv:2301.02909.

[1] Shared first author.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
DTAI
Celestijnenlaan 200A box 2402
B-3001 Leuven
lorenzo.perini@kuleuven.be
https://wms.cs.kuleuven.be/dtai