KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Analyzing scarce sensor data with applications in fleet-based anomaly detection and learning to reject

Kilian Hendrickx

Supervisors:
Prof. dr. ir. J. Davis
Prof. dr. ir. K. Gryllias
Dr. ir. B. Cornelis
(Siemens Industry Software)

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

March 2022

Analyzing scarce sensor data with applications in fleet-based anomaly detection and learning to reject

Kilian HENDRICKX

Examination committee:
Prof. dr. ir. B. Puers, chair
Prof. dr. ir. J. Davis, supervisor
Prof. dr. ir. K. Gryllias, supervisor
Dr. ir. B. Cornelis, supervisor
(Siemens Industry Software)
Prof. dr. ir. H. Blockeel
Prof. dr. ir. J. Suykens
Dr. ir. W. Meert
Dr. E. Tsiporkova
(Sirris)
Prof. dr. W. Waegeman
(UGent)

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

March 2022

© 2022 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Kilian Hendrickx, Celestijnenlaan 200A box 2402, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Acknowledgements

As a kid, my room was always stuffed with custom Lego builds, protected by my personal burglary system that I crafted together with some paperclips and an old buzzer. Combining my imagination with technology is something I've enjoyed ever since. Doing a PhD gave me the opportunity to regularly relive such inventive moments.

Although some see a PhD as a solitary journey, I consider mine as a team effort. Several people have impacted my journey and helped me to overcome various challenges.

I'm very grateful for the support from my promotors and advisors. Six years ago, Jesse suggested me to come to Leuven, to enroll in the MAI program, and start a PhD with him. Listening to that advice has been one of my best decisions ever made. I've learned a lot from his direct feedback style and greatly appreciated working with him. He's incredibly committed to his students, is always ready to help us, and manages to push me at exactly the right moments. Bram's pragmatic and constructive management style combined with his technical skills makes him more than just an industrial promoter. It's amazing how he links different technologies to tackle challenging engineering problems. I enjoyed our regular lunches and sports activities such as the research run and inter-company triathlon. and it was nice to see his ML research team mature over the years. Moreover, I was happy to have Kostis as my co-promotor. I still remember how we couldn't stop brainstorming when linking machine learning techniques to various applications. These sessions were really helpful to define the use-cases I've worked on. Moreover, his mechanical engineering expertise was of great value to me to understand the mechanical aspects in these use-cases. Finally, it was also a true pleasure to work with Wannes, whose advice was priceless for my PhD. I'm truly amazed by his technical insights and how he seems to manage fitting more than 24 hours in a day. In addition to his skills, he's just an amazing person to have around.

Reflecting on the past made me realize that some people affected my journey long before I started this PhD. Marc and Patricia steered me through my secondary education's graduation work in which they gave me my first research project without me realizing it. Giving me the freedom to define my own scope, letting me build a miniature wind tunnel, and constantly pushing my limits has definitely contributed to this work. Moreover, I would like to thank Anna, Elena, Mathias, Tom, and Vincent for their enthusiastic master thesis guidance. Our discussions during lunch, their passion for data science, and their excellent advice sparked my research interests. I vividly remember how you've opened a whole new world for me by sending me to an ICT in sports workshop. Without these experiences, I probably would have never started this journey.

During my master theses, I learned to appreciate working in applied research. I'm really happy that Bram and Karl gave me the opportunity to continue this journey with a PhD within Siemens. This was enabled by funding of the Flemish Agency for Innovation and Entrepreneurship (VLAIO), which excels in bringing academic and industrial research together. I'm grateful that they've supported me with a Baekeland mandate.

Being a member of the Siemens research team with Bart, Bart, Brecht, Carina, Claudio, Christophe, Deepti, Emilio, Enrico, Jacques, Ludo, Fabio, Frederico, Giampierro, Giancarlo, Mahmoud, Mathieu, Mariano, Steffen, Thomas, and Umberto was a unique experience. You've guided me while I struggled to understand various mechanical engineering applications and signal processing techniques. Moreover, Bram, Carina, Mathieu, Giampiero, Thomas, and Yannick played a critical role in my data collection campaigns. I still remember our times in the ULB lab with fresh Pasteis de nata and pasta cups, struggling with noise in the Siemens basement, as well working in the garage dismounting the interior of a car. I'm very grateful for your support and company during those days! Moreover, I'm happy to have had Thomas during our adventures in the student's office, enriched with *casino's* and ping-pong.

I greatly enjoyed being part of DTAI's ML research group. Even though I was only around on Mondays, they made me feel at home in the computer science department. Our lunches, hallway chats, and 4 pm coffee/cookie breaks were valuable distractions to me. Moreover, it was nice to see the enthusiasm for DTAI running team. Arne, Gust, Jessa, Jonas, Laurens, Robin, Toon, Tom, and Stella, I've enjoyed our weekly runs. Let's hope we can soon restore the "excuses of DTAI" into its original glory. Laurens, Jonas, and Evgeniya, thanks for hosting this 03.159 intruder. Stella, I'm really happy you managed to get my name on the door of this magnificent office which has some great predecessors. You're so much more than just an amazing project coordinator, thanks for everything (even for keeping me from writing these acknowledgments). Moreover, I would like to apologize to Ruben for chatting too much with Jessa

during her work hours, which occasionally made her stay late. Nevertheless, thanks Jessa for those fun moments, being supportive at the right times, and eventually returning my bicycle lights. I hope that these references are great additions to your acknowledgment collection. Collaborating with Dries, Jesse, Lorenzo, and Wannes on our machine learning with rejection work was one of my research highlights. Although having to rely on Teams meetings didn't make it easy to understand and unify five perspectives, I truly believe that our lengthy meetings led to many interesting insights and excellent work. Additionally, it made me forgive Lorenzo for kicking Belgium out of Euro 2021.

I'm thankful to all members of my jury to attribute their valuable time to my work. Their comments have significantly improved this dissertation. Elena, I was really happy to have you as a jury member as I greatly value your insights and experience with applying research in industrial settings. Willem, your expertise in uncertainty in AI was extremely valuable for this work. I'm sure your feedback will help to improve our survey paper even more in the future. Johan, I would like to thank you for your helpful feedback and excellent suggestions, as a juror as well as a member of my supervision committee. Straffe Hendrik, our hallway conversations, walks, and drinks made you more than a jury to me and I'll be happy to share that bottle of Gin ;-). Lastly, Bob, thank you for your advice and chairing my defense.

When arriving in Leuven, I quickly learned that the university is much more than its educational and academic setting. The KU Leuven Triathlon Team integrated me within my first few weeks and many new friendships arose ever since, rooted in Cédric and Glenn's coaching. In addition to the mutual suffering, I enjoyed the post-workout activities (of which some memorable footage still exists ;-)), beer miles (for which luckily only memories and results remain), legendary Hawaii nights (in which we rarely saw the finish), New Year parties (it doesn't take much alcohol for triathletes to start singing), and a trip to Verona (greatly hosted by Luca and Isabella). Moreover, I've great memories of our yearly training camps. Even though no KOMs remain for me, I'll always keep our tan lines, bike-dancing, and uphill sprints in mind! Thanks Arno, Brecht, Brecht, Cedric, Emma, Elke, Elise, Glenn, Hannah, Igor, Jef, Jessica, Joachim, Johan, Jolien, Jonie, Liese, Lore, Margot, Marica, Maura, Maurine, Matthias, Michiel, Pieter, Roxanne, Sarah, Silke, Stan, Stijn, Tim, Tim, and Toon!

It has been a true blessing to have over the years Brent, Hugo, Joris, Katinka, Lieze, Nand, Robbe, Sander, and Seppe around as roommates. The upgrade from T40 to Knalbond gave rise to whole new relaxing adventures such as the legendary Klub Knalbond and dress-up parties. Moreover, it was nice to have delicious dinners being prepared for me during busy periods!

Having spent the last decade in Ghent and Leuven implied that I couldn't

meet up with some of my friends as regularly as I wished for. Nevertheless, I greatly appreciate the occasions when we were able to catch up. Audrey, Daniel, Joaquim, Jonathan, Merel, and Nadine, it was amazing to have neighbors like you. I've always enjoyed our fence conversations, game nights, or casual no-reason visits, which typically lasted twice as long as planned. It's weird to be living more than 100 meters away from you, so let me know when your neighbors decide to move out ;-). Additionally, I'm always looking forward to a BBQ, birthday party, or another event with my former KAK-ers David, Fenna, Tim, and Roxanne. Let's hope we can reclaim the second place in our school quiz soon! Moreover, "Hoe laat?" won't ever have the same meaning thanks to Andreas, Cedric, Douwe, Fien, Hannah, Jamin, Koen, Manu, Sarah, Tim/Jos, and Vicky. Thanks for letting me spend your tax money and energize me during our trips to France, hikes and cycling tours, and annual weekends. Furthermore, I enjoyed having Andreas, Cedric, and Robbe around as ski/boarding buddies. I'm happy I've convinced at least one of you about the benefits of skiing and I look forward to the moment where each of you becomes independent on flat slopes. Finally, I would like to thank Charlotte, Jef, Michiel, Nicholas, Quentin, Robin, Silke, and Stijn. It's always a true pleasure to train, have spaghetti and drinks, or even renovate houses with you!

Verder wil ik ook mijn familie bedanken voor alle steun en kansen die ze me gaven. Papa en mama, merci om me de kans te geven mijn eigen weg uit te stippelen en me daarin altijd te blijven steunen. Kiara, bedankt voor alle taxiritjes, de telefoontjes als je je weer eens verveelt in de auto, en om je bed af te staan. Verder wil ik ook mijn grootouders bedanken. Al van kleins af aan kon ik altijd bij jullie terecht voor advies. Het was geweldig dat jullie elke schooldag opnieuw lekker eten hadden klaarstaan en ik genoot ook altijd van onze fietstochtjes, avonturen in de tuin, en uitstapjes.

Finally, Maura, thanks for all the happiness you bring and always being there for me. I'm really happy our paths crossed in the triathlon team. Thank you for our crazy moments, distracting me from work (even when you shouldn't), motivating me when needed, being who you are, and so much more.

Abstract

Smart devices and connected products have led to an increasing volume of operational product data. The wide availability of such data has led to various applications in domains such as smart healthcare, automotive, and industry. One commonly available type of operational data used in these applications is raw sensor data, which is collected from various measurement devices on or near the product. Analyzing large amounts of operational data requires intelligent and scalable analysis. Since manually coding such analysis is too time-consuming, automating various of its aspects with machine learning is appealing.

The adoption of machine learning in real-world applications remains challenging. This dissertation aims to overcome two challenges that often arise in practice. First, many machine learning techniques assume a wide availability of annotated data sets, containing examples of all relevant events for the application at hand. In reality, such data sets are often not available. Second, machine learning models might be inaccurate for data points it has not yet seen before. This is problematic for real-world applications, where such models should be robust to the occurrence of unexpected events. We tackle these challenges in two industrial-oriented applications: machine health monitoring and vehicle usage profiling.

Machine condition monitoring is the task of detecting mechanical issues before catastrophic failure. Several approaches based on machine learning have been proposed for this purpose. One approach trains a machine learning model to recognize patterns of machine failures. However, this requires an annotated data set of every possible failure, which is rarely available. Alternative approaches can help to detect a machine deviating from its expected healthy behavior, but this still requires a historical data set to define the expected behavior. Each of these approaches assesses machines individually. However, some industrial condition monitoring applications involve a fleet of similar operating machines. In most of these applications, it is safe to assume healthy conditions for the

majority of machines. Deviating machine behavior is then an indicator for a machine fault. We exploit this assumption by proposing a fleet-based condition monitoring framework that assesses a machine's health status by comparing its behavior with the other machines in the fleet. This tackles the aforementioned challenges, as our framework does not rely on historical data.

Vehicle usage profiling aims to derive insights into the driving habits of vehicle owners. Applying machine learning in this context is challenging. Machine learning models require representative training data, but it is infeasible for a vehicle manufacturer to predict all scenarios that real-world vehicles endure. This makes traditional machine learning models unreliable when used to process operational data collected in such unexpected scenarios. Instead, we study machine learning techniques that can detect unexpected vehicle usage, as well as abstaining from making a prediction in such scenarios. We tackle these challenges by using machine learning with a reject option. This machine learning subfield studies models that can abstain from making a prediction when likely to be incorrect.

This dissertation has three main contributions. First, it proposes a fleet-based condition monitoring framework that uses anomaly detection techniques to tackle the need for a historical data set. Second, it provides a literature survey on machine learning with a reject option which helps the further adoption of this subfield. Third, it demonstrates how vehicle usage profiling benefits from machine learning with a reject option.

Beknopte samenvatting

Slimme apparaten en geconnecteerde producten genereren grote hoeveelheden gebruiksdata. De brede beschikbaarheid van deze data leidt tot verscheidene applicaties in domeinen zoals slimme gezondheidszorg, automobiel en industrie. Meestal is in deze applicaties ruwe sensordata beschikbaar, welke gecollecteerd wordt door verschillende meettoestellen op of nabij het product. Het analyseren van grote hoeveelheden van zulke data vereist intelligente en schaalbare analyse. Het manueel coderen van deze analyse is echter te tijdrovend, waardoor het automatiseren van (aspecten van) deze taak door machine learning een interessant alternatief vormen.

Het blijft echter een uitdaging om machine learning toe te passen in praktische applicaties. Dit proefschrift tracht daarom twee vaak voorkomende uitdagingen te overwinnen. Ten eerste veronderstellen veel machine learning technieken een grote beschikbaarheid van geannoteerde datasets, welke voorbeelden van alle, voor de applicatie relevante, gebeurtenissen bevatten. In de realiteit zijn dit soort data sets vaak niet beschikbaar. Ten tweede kunnen machine learning-modellen onnauwkeurig zijn voor ongeziene datapunten. Dit is problematisch in praktische applicaties, waar dit soort modellen betrouwbaar moeten blijven in het geval van onverwachte gebeurtenissen. In die proefschrift lossen we deze uitdagingen op in twee industrieel-relevante toepassingen: foutdetectie van machines en gebruiksprofilering van voertuigen.

Foutdetectie van machines tracht (mechanische) problemen te detecteren voordat deze tot catastrofale gevolgen leiden. Verschillende benaderingen op basis van machine learning werden hiervoor voorgesteld. Een van deze benaderingen maakt gebruik van een machine learning-model dat in staat is om patronen van machinestoringen te herkennen. Het leren van dit model vereist echter een geannoteerde dataset van elke mogelijke storing, welke vaak niet beschikbaar is. Alternatieve benaderingen trachten afwijkingen van een machine te detecteren ten opzichte van verwachte gedrag. Het definiëren van dit verwachte gedrag vereist echter een historische dataset van verwacht machinegedrag in alle

mogelijke omstandigheden. Dit soort data is vaak niet voorhanden. Een gemeenschappelijke factor in beide benaderingen is dat deze de machines afzonderlijk beoordelen. Sommige industriële toepassingen omvatten echter een vloot van vergelijkbare werkende machines. In de meeste van deze toepassingen is het realistisch om te veronderstellen dat het merendeel van de machines in deze vloot geen defect vertoont. De afwijking van een machine ten opzichte van het merendeel van de vloot is dus een indicator voor een machinefout. We maken hiervan gebruik door een vlootgebaseerde aanpak voor foutdetectie voor te stellen. Deze aanpak beoordeeld de gezondheidsstatus van een machine door het gedrag ervan te vergelijken met de andere machines in de vloot. Dit heeft als grote voordeel dat dit soort aanpak niet afhankelijk is van historische gegevens, wat een belangrijke beperking is van bestaande technieken.

Gebruiksprofilering van voertuigen geeft inzicht in het rijgedrag van voertuigeigenaren. Het toepassen van machine learning in deze context is een uitdaging omdat machine learning-modellen representatieve trainingsgegevens vereisen. Het is echter onhaalbaar voor een voertuigfabrikant om alle gebeurtenissen te voorspellen die echte voertuigen doorstaan. Dit maakt traditionele machine learning-modellen onbetrouwbaar wanneer ze worden gebruikt om gebruiksdata te verwerken die bij dergelijke onverwachte gebeurtenissen zijn verzameld. We lossen deze uitdaging op door machine learning-technieken te bestuderen die in staat zijn om onverwacht voertuiggebruik te kunnen detecteren, en in zulke gevallen de mogelijkheid hebben om zich te onthouden van het maken van een voorspelling. We bestuderen hiervoor machine learning met een onthoudingsoptie. Dit deelgebied van machine learning bestudeert modellen die zich kunnen onthouden van het maken van een voorspelling wanneer deze waarschijnlijk onjuist is.

Dit proefschrift levert drie belangrijke bijdragen. Ten eerste stelt het een vlootgebaseerd aanpak voor foutdetectie voor, welke geen historische gegevens vereist dankzij het gebruik van anomaliedetectietechnieken. Ten tweede biedt het een literatuuronderzoek over machine learning met een onthoudingsoptie. Deze literatuurstudie zal helpen in de verdere ontwikkeling van dit deelgebied van machine learning. Ten derde laat het zien hoe gebruiksprofilering van voertuigen profiteert van dit soort machine learning-technieken.

List of Abbreviations

ANN Artificial Neural Networks. 10, 11

ARC Accuracy-Reject Curve. 90, 91

AUC Area Under the ROC Curve. 90, 91

AURC Area Under the ARC Curve. 91

CAN Controller Area Network. 124

DC Direct Current. 37–39

DL Deep Learning. 10

DTW Dynamic Time Warping. 14, 18–20, 42, 43, 52

FFT Fast Fourier Transformation. 17, 47, 52, 53, 57

FNR False Negative Rate. 90

FPR False Postive Rate. 100

GMM Gaussian Mixture Model. 87, 92, 105, 106

GPD Generalized Probabilistic Descent. 104

HMM Hidden Markov Model. 87

IRI International Roughness Index. 123

kNN k-Nearest Neighbors. 93–95, 113

kNNO k-Nearest Neighbors Outlier. 106, 122, 126, 128, 130, 140

KPI Key Performance Indicator. 117, 118, 120, 121, 123, 137

LOF Local Outlier Factor. 86, 106, 107, 109

LVQ Learning Vector Quantization. 108, 110

MAE Mean Absolute Error. 126, 128, 130

MCSA Motor Current Signature Analysis. 38

MSE Mean Squared Error. 128

OCSVM One-Class Support Vector Machine. 88, 92, 98, 106, 107, 122, 126, 128, 130, 140

PR Precision-Recall. 89

RBFN Radial Basis Function Network. 98

RF Random Forest. 125, 126, 130

ROC Receiver Operating Characteristic. 89, 91, 100, 101, 103

SCIM Squirrel cage induction motor. 37–39

SGD Stochastic Gradient Descent. 104

SMO Sequential Minimal Optimization. 104

SOM Self-Organizing Map. 96

SVDD Support Vector Data Descriptor. 92

SVM Support Vector Machine. 10, 85, 88, 89, 95, 97, 98, 104, 107–110

TPR True Postive Rate. 90, 100

VAE Variational Autoencoders. 87

WRSM Wound Rotor Synchronous Motor. 37–39

Contents

Abstract v

Beknopte samenvatting vii

List of Abbreviations x

Contents xi

List of Figures xv

List of Tables xvii

1 Introduction 1
- 1.1 Operational data analysis for industrial applications 3
  - 1.1.1 Machine health monitoring 4
  - 1.1.2 Vehicle usage profiling 5
- 1.2 Dissertation Statement 6
- 1.3 Contributions 6
- 1.4 Structure of the thesis 7

2 Framework for fleet-based anomaly detection 13
- 2.1 Introduction 14
- 2.2 General fleet-based condition monitoring framework 15
  - 2.2.1 Building block 1: Comparing two machines 16
  - 2.2.2 Building block 2: Clustering the machines in the fleet . 21
  - 2.2.3 Building block 3: Anomaly detection 23
  - 2.2.4 Building block 4: Visualization 25
- 2.3 Experiments 26
  - 2.3.1 Experiment 1: A fleet with malfunctioning machines . . 27
  - 2.3.2 Experiment 2: A fleet of gradual degrading machines . . 29
  - 2.3.3 Experiment 3: A fleet with a limited number of machines 30

2.3.4 Discussion . . . 32
2.4 Conclusion . . . 33

3 Applications of fleet-based anomaly detection 35
3.1 Introduction . . . 36
3.2 Application 1: Machine health monitoring through electrical signature analysis . . . 37
3.2.1 Experimental fleet test rig and data set . . . 37
3.2.2 Classic voltage unbalance diagnosis techniques . . . 38
3.2.3 Variant 1: Time domain . . . 42
3.2.4 Variant 2: Frequency domain . . . 47
3.2.5 Variant 3: Time/Frequency domain . . . 50
3.3 Application 2: Machine health monitoring through vibration signature analysis . . . 53
3.3.1 Experimental fleet test rig and data set . . . 55
3.3.2 Classic voltage unbalance diagnosis techniques . . . 56
3.3.3 Variant 1: Frequency domain . . . 57
3.4 Application 3: End-of-production-line testing . . . 61
3.4.1 Experimental setup . . . 62
3.4.2 Fleet monitoring framework for end-of-production-line testing . . . 63
3.4.3 Block 1: Gear comparison . . . 63
3.4.4 Block 2–4: Fleet clustering, Anomaly Detection & Visualization . . . 64
3.4.5 Experiment 1: Applicability for end-of-production-line testing . . . 64
3.4.6 Experiment 2: Size of historical data set for warm start 65
3.5 Conclusion . . . 66

4 Survey on machine learning with a reject option 75
4.1 Introduction . . . 75
4.2 Preliminaries on machine learning with a reject option . . . 76
4.3 Rejection types . . . 78
4.3.1 Ambiguity Rejection . . . 78
4.3.2 Novelty Rejection . . . 79
4.4 Properties of a rejection model . . . 80
4.4.1 Architectures for a model with a reject option . . . 81
4.4.2 Learning perspective . . . 84
4.4.3 Relation between model types and rejection . . . 86
4.5 Evaluating models with a reject option . . . 89
4.5.1 Evaluating models with fixed rejection rate . . . 90
4.5.2 Evaluating the model's accuracy/coverage trade-off . . . 90
4.6 Instantiating a model with a reject architecture . . . 91

4.6.1 Separated rejector . . . 91
4.6.2 Dependent rejector . . . 92
4.6.3 Integrated rejector . . . 97
4.6.4 Combining multiple rejectors . . . 99
4.7 Learning a model with a reject option . . . 99
4.7.1 Sequential learning . . . 99
4.7.2 Simultaneous learning . . . 102
4.8 Relation between model types and rejection . . . 104
4.8.1 Generative models . . . 105
4.8.2 Discriminative models . . . 107
4.9 Applications of machine learning models that reject . . . 108
4.10 Link to other research areas . . . 110
4.10.1 Uncertainty quantification . . . 110
4.10.2 Anomaly detection . . . 111
4.10.3 Active learning . . . 112
4.10.4 Class-incremental / incomplete learning . . . 112
4.11 Discussion . . . 113
4.11.1 Research questions revisited . . . 113

5 Machine learning with a reject option for usage profiling 117
5.1 Introduction . . . 117
5.2 Background on vehicle engineering: need for usage profiling . . . 119
5.3 Usage profiling: data science challenge . . . 120
5.4 Our approach for vehicle usage profiling . . . 120
5.4.1 Predictor . . . 121
5.4.2 Rejector . . . 121
5.4.3 Combined model . . . 122
5.5 Use-case: road-roughness analysis . . . 123
5.5.1 Data Collection and Preprocessing . . . 124
5.5.2 Experimental Methodology . . . 125
5.5.3 Results . . . 126
5.5.4 Discussion . . . 130

6 General discussion 133
6.1 Summary contributions and future directions . . . 133
6.1.1 Contribution 1: Framework for fleet-based condition monitoring . . . 134
6.1.2 Contribution 2: Literature survey on machine learning with a reject option . . . 135
6.1.3 Contribution 3: Robust machine learning for vehicle engineering . . . 137
6.2 Lessons learned for data scientists . . . 138
6.2.1 Data does not come for free . . . 138

6.2.2 Domain knowledge is crucial 139
6.2.3 Can I trust you? 140
6.2.4 Divide and conquer 140

Bibliography **141**

Curriculum vitae **177**

List of publications **179**
6.3 Journal papers 179
6.4 Conference papers 179
6.5 Papers under review 180

List of Figures

2.1 Graphical representation of the fleet-based condition monitoring framework . . . 16
2.2 Illustration of Dynamic Time Warping . . . 20
2.3 Motivation for Ψ-Dynamic Time Warping for sine waves . . . 20
2.4 Example of a dendrogram . . . 22
2.5 Intuition for cluster partitioning based on cophenetic correlation 23
2.6 Example visualization of the fleet-based condition monitoring framework . . . 26
2.7 Comparison between cluster-based anomaly detection and similarity-based anomaly detection . . . 28
2.8 Performance of the anomaly detection variants for a fleet with gradually degradation of machines . . . 31

3.1 Illustration of the experimental fleet setup . . . 38
3.2 Voltage unbalance based on current amplitudes . . . 40
3.3 Voltage unbalance based on current waveforms . . . 40
3.4 Voltage unbalance based on harmonics . . . 42
3.5 Dynamic time warping of healthy and faulty machines . . . 43
3.6 Cluster-based anomaly scores for a gradual degrading machine 46
3.7 Similarity-based anomaly scores for a gradual degrading machine 46
3.8 Visualization of our framework for frequency-based analysis . . 52
3.9 Visualization of our framework for time/frequency-based analysis 55
3.10 Drive train instrumentation for vibration signature analysis . . 56
3.11 Voltage unbalance detection based on harmonics of the acceleration signal . . . 57
3.12 Visualization of our framework for vibration signature analysis . 61
3.13 Visualization of our framework for end-of-production-line testing 65

4.1 Illustration of ambiguity and novelty rejection for classification 80
4.2 Illustration of ambiguity and novelty rejection for regression . . . 81

4.3 Illustration of a separated rejector 82
4.4 Illustration of a dependent rejector 83
4.5 Illustration of a integrated rejector 84

5.1 Instrumentation for road-roughness measurements 125
5.2 Results of our machine learning with a reject methodology for road roughness estimation 129
5.3 Novel road surfaces detected by our methodology 131

List of Tables

2.1 Simulated machine degradation 29
2.2 Minimal fleet size required by our condition monitoring framework 32

3.1 Possible prediction mistakes 37
3.2 Parameters of our experimental setup's motors 39
3.3 Results of classic voltage unbalance detection 41
3.4 Results of time-domain voltage unbalance detection using our framework 48
3.5 Results of time-domain voltage unbalance detection using our framework without intentionally introduced degrading in the drive train 49
3.6 Results of frequency-domain voltage unbalance detection using our framework 51
3.7 Results of time/frequency-domain voltage unbalance detection using our framework 54
3.8 Results of vibration signature analysis with a traditional technique 58
3.9 Results of vibration signature analysis for voltage unbalance detection using our framework 60
3.10 Gear design parameters. 63
3.11 Statistical features used for gear fault detection 67
3.12 Results of gear fault detection using our framework 68
3.13 Minimal fleet size required by our condition monitoring framework for gear fault detection 69

4.1 Notation used in the survey on machine learning with a reject option 78
4.2 Overview of rejection types that can be performed by different model types 87

5.1 Scenarios measured for the road roughness estimation training set 125

5.2 Sensors used for road-roughness estimation 126
5.3 Statistical features used for road-roughness estimation 127
5.4 Performance of several regression models for road-roughness estimation 128
5.5 Improvement added by the reject option for a model with 95% coverage 130

Chapter 1

Introduction

The Internet of Things and the reduced cost of data acquisition have led to an increasing amount of products with connectivity capabilities. Such products are both oriented to consumers and professionals, for example, wearables, connected vehicles, and smart machines. The connectivity capabilities of these products enable remote access to data collected during the product's operation, i.e., operational data. Different types of operational data exist. For connected products, two important data types are:

Raw sensor data Sensor data is collected from various measurement devices installed on or near the product. Examples of measurement devices are accelerometers, microphones, and thermometers. Such sensors require samples rates from less than 1 Hz to more than 20 kHz.

Edge processed data Products with computing abilities such as a CPU or microcontroller can process operational data on the device itself, which is called edge processing. By only transferring processed information specific to the application at hand, edge processing limits the required network bandwidth. For example, sensors that have a high sample rate produce vast amounts of data. Instead of transferring all the data collected by the sensor, statistical summaries (such as average or variance) or domain-specific features can be computed and transmitted. Alternatively, edge processing can extract predefined events from the operational data into event log files.

The wide availability of operational data is embraced by both industry and the academic community and has led to various innovative applications [1, 2]. Some

examples of such application domains are:

Smart health-care In the past, patients needed to be hospitalized or regularly visit a doctor to closely monitor their physiological status. Nowadays, smart watches [3], fitness trackers [4], and biosensors [5] have enabled continuous remote patient monitoring, improving health care quality [6]. Moreover, some of these technologies have found their way to consumer markets and are embedded in products such as sport gadgets [7].

Traffic monitoring Traffic flow was previously measured by induction loops installed in roads [8] and manual counting [9]. However, these techniques lack the real-time insights required for traffic jam information and prevention. More recently, smartphone data is utilized to have real-time insights into road occupation, traffic speeds, and incidents [10].

Agriculture Devices such as soil moisture and temperature sensors allow farmers to rely on real-time data to manage their crop water levels and conserve water [11]. More recently, devices such as agricultural drones assessing crop growth and managing insect control are gaining popularity in an attempt to increase efficiency [12].

Logistics Instead of manual bookkeeping, tracking of logistical goods is automated by positional sensors and techniques such as RFID [13]. These technologies help to optimize planning, improve operational processes, and reduce the amount of lost goods [14, 15].

Industry Operational data is heavily used in industry, ranging from energy management [16] to supporting manufacturing processes [17]. Companies such as Siemens (Mindsphere[1]), GE (GE Digital[2]), and Microsoft (Azure IoT[3]) offer various products and services for this purpose. The adoption and popularity of these have even led to the introduction of terms such as Industrial Internet of Things [18] and Industry 4.0, referring to the fourth Industrial Revolution [19]. Intelligent monitoring and data-driven decisions are deployed to increase efficiency, improve productivity, and transform the workplace [20].

[1] https://siemens.mindsphere.io/en
[2] https://www.ge.com/digital/
[3] https://azure.microsoft.com/nl-nl/overview/iot/

1.1 Operational data analysis for industrial applications

Processing vast amounts of operational sensor data requires intelligent and scalable analysis. Manually coding patterns and rules to perform tasks such as classification and forecasting is too time-consuming and does not scale well. Instead, machine learning is an appealing alternative to automate various aspects of the analysis. However, practitioners encounter several reoccurring challenges when applying machine learning to real-world applications such as:

Data availability The availability and quality of data sets are a common challenge in real-world applications. Some use-cases require this data set to contain all relevant events for the application at hand, which might be hard or even impossible to collect in applications such as machine failure prediction [21].

Data quality Real-world data sets can be incomplete, unlabeled, or incorrect. This might be problematic for some machine learning algorithms. Moreover, processing real-world data sets might have additional challenges. For example, machine failure prediction data sets can be imbalanced as failures might be relatively rare [22]. Additionally, the data might be prone to selection bias [23]. For example, a machine learning algorithm might falsely learn from hospital records that asthma has a positive effect on the pneumonia survival rate. In reality, this increased survival rate originates from directly admitting asthma patients to the intensive care unit [24].

Data preprocessing The performance of a machine learning pipeline highly depends on the data preprocessing quality [25]. For example, the quality of used features drastically impacts the model's predictive performance [26]. This even applies to machine learning techniques that aim to automatically generate these features, such as Neural Networks [22].

Unexpected data Data presented to models deployed in real-world applications might originate from other distributions than the data used to train these models [27]. As the model might be inaccurate for such unexpected data points, correctly processing these is an important challenge.

In this dissertation, we focus on processing operational sensor data and aim to overcome two challenges related to data scarcity that often arise in practice: i) the lack of historical (labeled) data that can be used to train a machine-learned model, and ii) when deployed, a learned model encounters situations or events

that were not present in the training set and hence the model's predictions are unreliable in these situations. More specifically, we tackle these challenges in two industrial-oriented applications: machine health monitoring and vehicle usage profiling.

1.1.1 Machine health monitoring

Detecting machine faults before catastrophic failure is essential for avoiding severe consequences and optimizing production up-time. In the past, such machine condition monitoring was typically automated by handcrafted rules provided by domain experts [28, 29, 30]. As these rules are often specific to a certain type of machine and operational condition, such a handcrafted approach does not scale well [31].

Supervised machine learning has been suggested to overcome this challenge, using a training set to recognize a predefined set of product failure patterns [32, 33]. However, collecting such a training set is hard as it requires the failure of several machines and prior knowledge of every possible mechanical failure [34, 35]. Semi-supervised machine learning such as novelty detection is an interesting alternative, but still requires a historical data set of healthy machine behavior in each possible operational condition [36, 37, 38, 39]. For novel machines, such a data set might not be available.

Industrial environments often deploy a fleet of similar operating machines [40, 41, 42, 43]. For example, factories operate multiple parallel production lines in which machines perform identical tasks. Alternatively, wind turbines in the same geographical location are prone to similar environmental conditions and expected to behave similarly. We exploit this phenomenon and tackle the aforementioned challenges by employing fleet-based condition monitoring, for which we make three reasonable assumptions:

1. In a fleet of machines, all machines in similar operational conditions have similar signatures;
2. The fleet consists of similar machines and the majority of these machines are operating normally (i.e., are in a healthy state);
3. If a machine's signature deviates from those of the majority, this indicates a fault such as a failing component.

Making these assumptions allows altering what data is required to perform automated anomaly detection. Instead of the traditional approach of predicting a machine's health status based on its historical data, our framework allows

assessing its status by comparing its behavior with the current behavior of the other machines in the fleet. This approach does not rely on historical data while still offering the capability of detecting any machine fault that affects the operational data, even if such a fault was not foreseen or even never occurred in the past.

In this dissertation, we propose a general fleet-based condition monitoring framework based on anomaly detection and study its applicability in two real-world applications: operational testing of a fleet of electrical machines and end-of-production-line testing for gears. Our objective is to detect faults without the need for a historical data set.

1.1.2 Vehicle usage profiling

Sustainable vehicle development aims to address shortcomings to existing vehicles in future design iterations. Consequently, a thorough understanding of these shortcomings and the customer's driving habits are key [44]. In the past, these insights were derived from questionnaires, expert analysis, or dedicated test campaigns [45, 46]. However, this is hard to carry out and does not scale well.

Recently, manufacturers start to overcome this challenge by the introduction of connected cars [47]. In addition to innovative services such as home integration, vehicle tracking, and on-road assistance, it also allows them to monitor the real-world usage of their production vehicles [48, 47, 49]. This monitoring enables manufacturers to tailor their vehicles to the customer's needs.

Vehicle manufacturers analyze operational vehicle data with supervised machine learning for applications such as autonomous driving (e.g., object recognition [50]), fleet monitoring (e.g., fuel consumption prediction [51]), and control systems (e.g., adaptive power control [52]). However, many of these algorithms assume the operational data to be similar to the training data. This implies that such machine learning models are only accurate in scenarios foreseen in the training data. If a customer uses the vehicle in a novel, unexpected scenario, the model's predictions are unreliable as its assumptions are voided.

In this dissertation, we aim to design a machine learning model that is robust to novel vehicle usage and provides reliable predictions for expected scenarios. Our objective is to gain accurate insights into the road profiles encountered by a test vehicle and to detect unexpected road types. For this purpose, we study the applicability of machine learning with a reject option to vehicle engineering in a road-roughness study. Such machine-learned models are aware of when

they are at risk of making a misprediction and have the ability to abstain from doing so in these situations.

1.2 Dissertation Statement

This dissertation aims to address two challenges related to data scarcity that pose obstacles to using data science techniques in real-world applications. First, we tackle the need to have access to a sufficiently large historical database to train a machine learning model for condition monitoring. Second, we study how vehicle usage profiling benefits from machine learning techniques that are robust to novel data points by abstaining from making a prediction in situations where they are likely to be incorrect. More specifically, we study the following research questions:

- How does exploiting the fleet allows detecting machine failures without requiring historical data?
- How can machine learning with a reject option make vehicle usage profiling prediction robust to unforeseen vehicle usage
- Can machine learning models with a reject option help to detect novel vehicle usage?

1.3 Contributions

To summarize, the main contributions of this thesis are:

Framework for fleet-based condition monitoring We provide a novel framework for fleet-based condition monitoring based on anomaly detection techniques. The applicability of this framework is demonstrated on two use-cases: an operational testing of electrical machines and end-of-production-line testing of gears.

Literature survey Machine learning with a reject option is quickly gaining attention from both machine learning researchers and practitioners. However, much of the research is disconnected and uses non-uniform terminology. We contribute to the further adoption of this subfield by presenting a structured overview of this domain.

Robust machine learning for vehicle engineering We demonstrate the benefits of machine learning with a reject option in a vehicle usage profiling application. Through a road roughness study, we show that the reject option yields more accurate insights in vehicle usage and allows the identification of novel road types.

1.4 Structure of the thesis

This dissertation is structured in two parts:

- The first part is oriented to fleet-based condition monitoring. We introduce a framework for fleet-based condition monitoring in Chapter 2 and demonstrate the applicability of this framework to real-world applications in Chapter 3.
- The second part of this dissertation is focused on detecting novel vehicle usage based on machine learning with a reject option. We provide an in-depth survey on machine learning with a reject option in Chapter 4. Next, we demonstrate its applicability for vehicle engineering in Chapter 5.

Finally, Chapter 6 reflects on this research by providing the lessons learned, general conclusions, and pointers to potential directions for future research.

Part I: Anomaly detection for fleet-based condition monitoring

This introduction is based on:
K. Hendrickx, W. Meert, Y. Mollet, J. Gyselinck, B. Cornelis, K. Gryllias, and J. Davis. A general anomaly detection framework for fleet-based condition monitoring of machines. *Mechanical Systems and Signal Processing*, 139:106585, may 2020. ISSN 08883270. doi: 10.1016/j.ymssp.2019.106585

Optimal performance and reliability of machines are important for several applications and industries, as in certain circumstances failures or malfunctions may lead to high production losses, severe injuries, or even loss of lives. Condition monitoring allows for the early detection of machine failures, which helps to avoid further damage and/or to reduce downtime.

The traditional condition monitoring approach involves handcrafting an indicator for the phenomena of interest, which requires insights into the expected machine usage and significant expertise in machine signature analysis. Once such a monitoring system is deployed, the current value of the indicator is compared to a pre-selected threshold to decide on the machine's health condition. In practice, this approach has several drawbacks. First, machine behavior can be influenced by operational conditions, which requires either setting different thresholds [54, 55], identifying dedicated operational conditions for fault detection (e.g. ISO 10816-3:2009 [56]) or normalizing the fault indicator [31]. None of these approaches are scalable to a large number of operational conditions. Second, a machine can be sensitive to multiple fault types. Thus, traditional condition monitoring approaches require a manual tuning procedure for each

expected machine fault, as well as combinations thereof. For complex systems, it is often not possible to know all these faults and combinations upfront. For this reason, an alternative approach is to compare operational machine data with a simulation model of healthy behavior [57]. However, such models can be hard to generate and this approach assumes that no variables change that might alter a health machine's behavior. This is hard to guarantee in reality, for example when environmental parameters vary.

Recent condition monitoring studies often suggest employing techniques from artificial intelligence as an alternative to these handcrafted solutions. In particular, the two most commonly mentioned subfields of AI are machine learning and data mining [54, 58, 59, 60]. Typically, researchers consider supervised, semi-supervised and unsupervised techniques from these two subfields.

Supervised approaches to fault detection such as Artificial Neural Networks (ANN) [61, 62, 63], Deep Learning (DL) [60, 64, 65, 66] and Support Vector Machine (SVM) [67, 68, 32] require large historical labeled data sets. That is, the data must be annotated with the true machine health condition. Moreover, they often assume the data contains examples of every possible fault type [69]. However, such data sets are not always available in industrial environments [70, 71, 21]. In those applications, even a limited historical data set such as the [72] proposes can be challenging to obtain. As an alternative, a physical model can simulate faults [32, 73], but this is often very challenging in practice as not every possible fault can be accurately modeled.

To alleviate the need to collect data about faults, another approach is to consider one-class methods like novelty detection [74, 75]. These techniques simply use a historical data set containing only healthy machines to build a model of healthy behavior. When deployed, they detect behaviors that deviate compared to the model of normal behavior [76, 77, 78]. Consequently, it is very important that data from a faulty machine is not present in the training data. Unfortunately, this is hard to guarantee in reality as machines degrade over time. Moreover, this assumes that normal behavior does not change over time, which may not be the case. An alternative approach is to cluster historical data and assign a label to each cluster. This allows classifying new machine measurements by assigning it the label of its nearest cluster. However, this approach requires expert knowledge to provide the labels [79].

Unsupervised anomaly detection is suggested to tackle the drawbacks of the supervised and semi-supervised approaches [80, 81, 82, 83]. These techniques require neither labeled data nor solely healthy behavior. Instead, unsupervised techniques assume that the majority of the data set belongs to the normal (healthy) class and identify rare behaviors as abnormal (faulty). Still, using

these techniques poses two main challenges in practice. First, in a condition monitoring context, it is often necessary to collect an anomaly detection data set of machines in similar operational conditions, especially when a historical data set is not available. Second, a drawback shared by many of the existing approaches is that they employ black-box models, and hence it is difficult to gain insight into why a model made a certain prediction. Consequently, domain experts may not trust the predictions that the model is making. Moreover, it is non-trivial to incorporate expert knowledge into models such as ANNs.

We tackle these challenges by monitoring multiple similar machines or entities (a fleet) simultaneously. While classical condition monitoring approaches are limited to analyzing a single machine [84], fleet-monitoring is advantageous in several real-world industrial applications [40, 85]. For example, this need arises in wind turbines [54], production lines, and aerospace engines (which often operate in pairs) [86]. Moreover, in fleet settings like monitoring the components of a train or a conveyor belt (such as the traction motors, bearings, and bogies [87, 88]), the machines in the fleet are operating under similar conditions.

In each of the previous examples, the different entities have comparable signature characteristics which deviate when a fault appears. We leverage this deviation and consider it as an indicator of machine failure. More specifically, Chapter 2 introduces a general condition monitoring framework for fleet-based machine monitoring. Chapter 3 presents results when applying the framework for operational and end-of-production-line monitoring.

Chapter 2

Framework for fleet-based anomaly detection

This chapter is based on:

- K. Hendrickx, W. Meert, Y. Mollet, J. Gyselinck, B. Cornelis, K. Gryllias, and J. Davis. A general anomaly detection framework for fleet-based condition monitoring of machines. *Mechanical Systems and Signal Processing*, 139:106585, may 2020. ISSN 08883270. doi: 10.1016/j.ymssp.2019.106585
- K. Hendrickx, W. Meert, B. Cornelis, K. Gryllias, and J. Davis. Similarity-based anomaly score for fleet-based condition monitoring. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society, PHM*. PHM Society, 2020
- K. Hendrickx, W. Meert, J. P. D. C. Patricio, B. Cornelis, K. Gryllias, and J. Davis. Fleet-based health monitoring for end-of-production-line and operational testing. In *Online Proceedings of ISMA2020*, Leuven, Belgium, 2020. KU Leuven

2.1 Introduction

This chapter proposes a general fleet-based health monitoring framework for condition monitoring that only relies on instantaneously available data and does not require a historical data set. The framework uses clustering and anomaly detection techniques to simultaneously analyze a fleet of machines and exploits the assumption that each machine is in a comparable operational condition to identify anomalous behavior.

To perform the analysis, the framework uses three inter-related generic blocks: comparing pairs of machines, clustering the entire fleet of machines, and identifying anomalous machines. Each block should have a meaningful outcome that can be visualized to give a domain expert an overview related to the observed machine behavior. First, we discuss the machine comparison block and propose several different ways to compare pairs of machines. Most interestingly, we consider a distance measure based on the amount of warping that occurs in a dynamic time warping. While measures based on Dynamic Time Warping (DTW) are common in the AI literature, these receive less attention in the context of mechanics and condition monitoring. Second, we employ clustering, a core artificial intelligence technique [91]. While the clustering can be done with a variety of algorithms, we recommend using a hierarchical method because of its simplicity and interpretability. We then propose an approach to partition a hierarchical clustering into a flat one that is novel within the context of machinery diagnostics. Finally, to identify potentially anomalous machines, the anomaly detection block exploits the reasonable assumption that the majority of the machines in the fleet are healthy. Consequently, a good indication of a fault is when a machine's current behavior deviates from the behavior currently exhibited by the majority of the machines. Like all condition monitoring and anomaly detection applications, this assumes that a fault alters the data generated by a machine [92].

The proposed methodology offers four advantages over existing machine learning-based condition monitoring approaches. First, unlike other approaches, the framework does not require any historical data. Instead, it makes use of online comparisons among machines. However, if they are available, recent machine measurements can improve performance by extending these comparisons. Second, the framework enables interpretable visualizations of the inter-related generic blocks. These allow a domain expert to gain insight into the framework's predictions. Hence, this dissertation considers relatively simple but intuitive techniques. Third, exploiting relevant domain knowledge allows tailoring the framework towards the specific use case. For example, a measure to compare a machine pair's behavior can be based on mechanical expertise. This is much harder with techniques such as neural networks and deep learning. Finally, the

framework can be applied directly to time-series data (i.e. raw measurements) and does not require a feature-based representation. However, the framework can use these features if available.

The remainder of this chapter is organized as follows. Section 2.2 proposes the fleet-based condition monitoring framework. Multiple possible configurations are discussed for each of the building blocks. Next, the framework is experimentally validated in Section 2.3. Finally, Section 2.4 discusses some general insights and conclusions.

2.2 General fleet-based condition monitoring framework

We propose an anomaly detection framework for fleet condition monitoring of similar machines, which detects deviating, thus probably faulty, machine behavior. This approach does not require a training data set and can detect various faults, even those not considered a priori. The fault detection process relies on three key assumptions:

1. In a fleet of machines, all machines in similar operational conditions have similar signatures;
2. The fleet consists of similar machines and the majority of these machines are operating normally (i.e., are in a healthy state);
3. If a machine's signature deviates from those of the majority, this indicates a fault such as a failing component.

Practical examples where all of these assumptions hold are applications such as monitoring a conveyor belt or railway components. In both of these cases, a deviating machine is an indication of a machine fault.

Three building blocks interact in the proposed framework: machine comparison, fleet clustering, anomaly detection. Additionally, these blocks can visually be presented to the user (Figure 2.1). The design and configurations of each block can be adapted to the application's needs, based on domain expertise. The first block measures the similarity between two machines by comparing their behaviors. A variety of measures can be used in this block. While we discuss several concrete possibilities, a user can select an appropriate measure for the application at hand based on domain knowledge. The second block is a clustering algorithm that uses the selected comparison measure to group

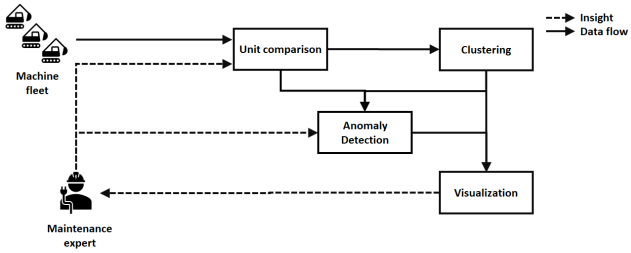


Figure 2.1: Interaction of the four blocks. Visualization can aid in interpreting the predictions. A user can select the machine comparison and anomaly score blocks based on domain knowledge, if available.

similarly behaving machines. The third block assesses the health status of a machine by assigning an anomaly score to each machine. The higher this score, the more deviating a machine's behavior is considered to be. In this work, we propose two anomaly detection approaches. Both approaches exploit our second assumption, implying that a faulty machine could only occur in a cluster that consists of less than half the machines. Finally, the results of these blocks can be visualized and presented to a user, providing a deeper understanding of the framework and guiding domain experts to analyze a specific deviating machine.

The framework performs online fleet analysis by considering small analysis windows. The required window size depends on the metrics selected in the first block. For example, evaluating in the frequency domain might require more data for better frequency resolution. As accidental outliers could affect predictions, the framework's robustness can be improved by considering a machine faulty based on n consecutive analysis windows.

2.2.1 Building block 1: Comparing two machines

Clustering uses the similarity of entities to infer a number of groups (clusters). Deciding how to compare a pair of machines (X, Y) is thus an important step in the proposed framework. Optionally, it allows exploiting engineering knowledge to guide the clustering. This block consists of two actions. The first action transforms the machine data into the desired signal domain. Secondly, the machines are compared in this new domain. Sequential data is represented as $(x_1, x_2, \ldots, x_n)$ where x_i is one data point and n is the length of the sequence. Each machine in the fleet is assumed to have the same total number of data points n.

Domain specific preprocessing

Operational machine data is generated by sensors that capture measurements in the time domain. Transforming the raw data to another signal domain can emphasize certain physical phenomena. The framework can support any type of transformation. This work focuses on four output domains: time, frequency, time/frequency, and feature space.

For all domains, it is important to normalize the data before comparing machines to help remove differences that simply arise due to the unique characteristics of each machine and cope with the fact that the measurement values can originate from different mechanical units. Without normalization, a comparison between machines would be dominated by the unit with the largest range.

Time domain Waveforms can be used to analyze time-domain measurements, as they contain information about amplitude, shape, and existing symmetries. A time-domain analysis is particularly interesting to detect trends and patterns over time.

Time series normalization is used to scale all machine measurements to the same range. Methods such as min-max scaling, $^{(X-min(X))}/_{(max(X)-min(X))}$, and z-score scaling, $^{(X-\overline{X})}/_{std(X)}$, are popular, but sensitive to outliers and should be used with care. If outliers are expected, outlier-robust estimators like percentile scaling, $(^{X-median(X))}/_{(\%ile_{75}(X)-\%ile_{25}(X))}$ (with $\%ile_{25}(X)$ the nth percentile of X), are preferred.

Frequency domain A Fast Fourier Transformation (FFT) can be used to convert time domain data to the frequency domain, which can yield insights that are hidden in the time domain. However, it is not possible to track trends and patterns over time. Techniques such as cyclostationary analysis [93] can help to analyze rotating machinery.

The previously mentioned normalization techniques are applicable in this domain as well. To remove differences due to the unique machine characteristics, mechanical insights could be of use. For example, frequency spectra can be scaled by a specific frequency corresponding with a mechanical phenomenon or use signal processing techniques to obtain equal Parseval energy.

Time/frequency domain The time/frequency domain detects frequency trends and patterns over time and can be shown in a spectrogram. Different normalization procedures can be applied, depending on the application's need.

One can normalize each of the obtained spectra using the techniques described in the frequency domain. Another option is to scale the complete spectrogram using the techniques from the time domain.

Feature space Various feature extraction methodologies, based on statistics, signal processing, or physics, can extract indicators for fault diagnostics from any signal domain. Examples are amplitude in the time domain, order extraction, or more complex signal processing techniques such as envelope analysis. The framework can use any of these indicators as a representation of machine behavior. In this case, normalization is required to equally weigh all features.

Dissimilarity measure

A measure[1] $s(X, Y)$ is needed to compare the distance or dissimilarity of a machine pair (X, Y), represented by sequences $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$ with length n. A data point x_i or y_i can be 1-dimensional, such as for the time domain, or m-dimensional, for example when representing a spectrogram or a feature space with m features.

Various functions to measure dissimilarity between sequences can be considered. An existing one can be selected using engineering knowledge or one can be designed specifically for the problem at hand. Two popular measures are:

1. Euclidean distance The Euclidean distance is the straight-line distance between two sequences of length n, given by:

$$s(X, Y) = \sqrt{\sum_{i=1}^{n} c(x_i, y_i)^2} \tag{2.1}$$

with $c(x_i, y_i)$ being the Euclidean norm between two m-dimensional data points:

$$c(x_i, y_i) = ||x_i - y_i||_2 = \sqrt{(x_{i,1} - y_{i,1})^2 + \cdots + (x_{i,m} - y_{i,m})^2} \tag{2.2}$$

A disadvantage of the Euclidean distance is its sensitivity to subtle shifts in the time series. This is solved by DTW.

[1] For the proposed configuration, a measure suffices. The triangle inequality, which must hold for a metric, is not required.

2. Dynamic Time Warping DTW is a successful algorithm in time series analysis that aligns in a nonlinear fashion a sequence representing time series, facilitating further analysis [94, 95]. The goal is to find an alignment of sequences X and Y that has the minimum total distance. The dissimilarity between X and Y is then expressed as:

$$s(X,Y) = \min_{p} \sqrt{\sum_{\theta=1}^{L} c\left(x_{p_{\theta,0}}, y_{p_{\theta,1}}\right)^2} \tag{2.3}$$

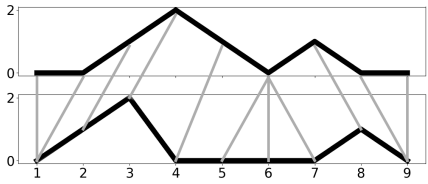
where $p = (p_1, p_2, \ldots, p_L)$ expresses an optimal alignment with length L, also referred to as the warping path. Each p_θ represents a tuple where the first element refers to the element in the x series on position $p_{\theta,0}$ and the second element refers to the element in the y series on the $p_{\theta,1}$ position. It satisfies three conditions:

1. $p_1 = (1,1)$ and $p_L = (n,n)$ (boundary condition)
2. $p_{1,0} \leq p_{2,0} \leq \ldots \leq p_{L,0}$ and $p_{1,1} \leq p_{2,1} \leq \ldots \leq p_{L,1}$ (monotonicity condition)
3. $p_{\theta+1} - p_\theta \in \{(1,0),(0,1),(1,1)\}$ for $1 \leq \theta \leq L-1$ (step size condition).

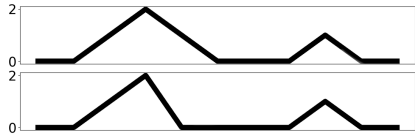
The dissimilarity of individual data points x_i and y_j is defined by a cost function $c(x_i, y_j)$, the Euclidean norm (Equation 2.2) by default. The sequences should be normalized before applying DTW, to avoid suboptimal alignment due to differences in amplitude. Figure 2.2 shows an example in which two arbitrary time series are aligned. Note that the Euclidean distance is considerably larger compared to the DTW dissimilarity even though the series are quite similar.

A variant to DTW used in this work is Ψ-DTW. Periodic signals can be misaligned due to a phase offset, which can result in DTW introducing undesired warping at the beginning and end (Figure 2.3a). This additional warping can lead to an unwanted increase in the DTW measure. Ψ-DTW [96] addresses this issue and avoids the undesired warping (Figure 2.3b). It does this by relaxing the DTW boundary condition by $p_1 = ([1,\Psi],1)$ or $p_1 = (1,[1,\Psi])$, and $p_L = ([n-\Psi,n],n)$ or $p_L = (n,[n-\Psi,n])$, where $\Psi \in \mathbb{N}$ is a user-defined parameter. Apart from this, it operates in a similar way as the classic DTW. The example in Figure 2.3 shows that for periodic signals, Ψ-DTW leads to a better match.

DTW warping amount (Ψ-)DTW is known to work well when it is important to compare time series based on their shape. However, in some cases, DTW

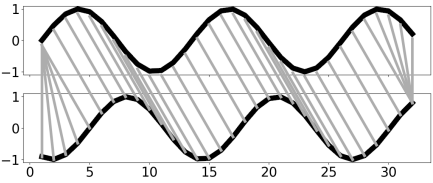


(a) Original time series.

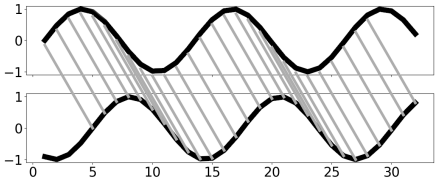


(b) Optimal aligned time series. The X-axis is hidden, as the time domain is warped.

Figure 2.2: DTW finds an optimal alignment between two time series (black) by means of a warping path (gray). Euclidean distance in (a) is 7.0, the non-linear DTW in (b) has a dissimilarity of 1.0.



(a) Classic DTW. Dissimilarity: 1.8



(b) Ψ-DTW. Dissimilarity: 0.0

Figure 2.3: DTW on sine waves with a phase shift. Warping paths are shown in gray. While identical, the sine waves are considered dissimilar by classic DTW due to additional warping in the beginning and end. Ψ-DTW removes this effect and correctly identifies the sine waves as identical.

removes the difference between healthy and faulty signals. One example is in electrical current waveforms which are considered in one of the applications. To overcome this challenge, a measure that considers the amount of warping that occurs in Ψ-DTW is used [97]:

$$s(X, Y) = \sum_{\theta=1}^{L-1} [p_{\theta+1} - p_\theta \neq (1, 1)]. \tag{2.4}$$

It is advised to use a normalized form, $s(X, Y)/L$, when using Ψ-DTW, as the length of the paths can vary.

2.2.2 Building block 2: Clustering the machines in the fleet

This block identifies similarly behaving machines by employing a clustering algorithm that uses the selected measure $s(X, Y)$ to compare pairs of machines. There are two important points to consider when selecting a clustering algorithm. First, the selected clustering algorithm should work for an arbitrary measure $s(X, Y)$. Some algorithms only work with certain measures. For example, *K-means* requires a space where the mean is defined. Therefore, it is preferable to use an algorithm that represents clusters by actual data points and is hence more generic, such as K-medoids [98, 99], spectral clustering [100], hierarchical clustering [101] and DB-scan [102]. Second, the number of clusters is unknown in most anomaly detection tasks. If all the machines are healthy, a single cluster is expected whereas multiple clusters are needed if there are faulty machines. Some methods, like K-medoids, require the user to specify the number of clusters k in advance. In contrast, approaches like hierarchical clustering and DB-scan derive the number of clusters from the structure of the data, often by using a parameter that describes how tight a cluster should be.

Hierarchical clustering is proposed as the default implementation for this block. It offers the benefits of being relatively simple, producing results that can be easily visualized, and not having to set k upfront.

Hierarchical clustering Hierarchical clustering builds a hierarchy of clusters. This can be done in either an agglomerative (bottom-up) or divisive (top-down) manner. However, agglomerative clustering is more common. This approach begins by assigning each item to its own subcluster. Then, the algorithm iteratively merges the two nearest (or most similar) subclusters until there is only one cluster left. The function to measure the distance between two subclusters is a parameter of hierarchical clustering. Two common choices are: *single linkage*, which considers the minimum distance between the elements of two subclusters, and *complete linkage*, which considers the maximum. Other alternatives are *average linkage* and Ward's method [103].

The result of the clustering can be represented in a tree-like structure called a dendrogram (Figure 2.4). It is relatively easy to interpret this structure, which is an additional benefit over other clustering techniques. At the lowest level, each cluster contains a single item. Going from the bottom-up, each subcluster is merged with its most similar subcluster until a single cluster is obtained. The vertical axis shows the distance between each of these subclusters. The higher this merge occurs, the less similar the subclusters are.

Some anomaly detection approaches require that each machine is assigned to only one cluster. The hierarchical structure of the clustering needs to

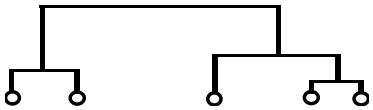


Figure 2.4: A dendrogram representing a cluster hierarchy. Agglomerative (bottom-up) clustering begins by assigning each item (circles) to its own cluster and iteratively merges the two most similar clusters until only one cluster is left. In this example, the horizontal distance between the items is considered to represent their distance.

be partitioned to satisfy this constraint. A naive approach is to cut the hierarchy by using a threshold on the distance between subclusters. Selecting an appropriate threshold depends on $s(X, Y)$ (for example its range) and does not take the underlying cluster structure into account. To overcome this challenge, a threshold on the cophenetic correlation thr_{cc} is proposed. This offers the advantages of being independent of the range of $s(X, Y)$ and considering the structure of the data.

The cophenetic correlation measures how well a clustering preserves the original pairwise distances [104], that is, how well the clustering reflects the original structure of the data. A high cophenetic correlation indicates that two or more cluster partitions are present in the data, while a low cophenetic correlation suggests that only a single cluster partition is present in the data. Formally, the cophenetic correlation is defined as Equation 2.5:

$$cc = \frac{\sum_{(X,Y)\in P}(s(X,Y) - \bar{s})(t(X,Y) - \bar{t})}{\sqrt{[\sum_{(X,Y)\in P}(s(X,Y) - \bar{s})^2][\sum_{(X,Y)\in P}(t(X,Y) - \bar{t})^2]}}. \tag{2.5}$$

where P is the set of all machine pairs (X, Y), $s(X, Y)$ is the measure used in the clustering, and $t(X, Y)$ is the dendrogrammic distance, which is the height at which machines X and Y are first joined in the dendrogram[2]. Finally, $\bar{s}$ and $\bar{t}$ are the average pairwise dissimilarities and dendogrammic distances. A pseudocode for splitting a dendrogram is given in Algorithm 1.

The intuition behind the cophenetic correlation partitioning procedure is illustrated with a simple example (Figure 2.5). First, the upper part considers a single cluster (shown in red). There is a difference between the pairwise and dendrogrammic distances. The correlation between them is rather low. Second, the lower part of the figure shows a situation where a second cluster is present (shown in blue). Now, the distances within a cluster are relatively small. In

[2]Note that the dendrogrammic distance depends on the selected linkage function.

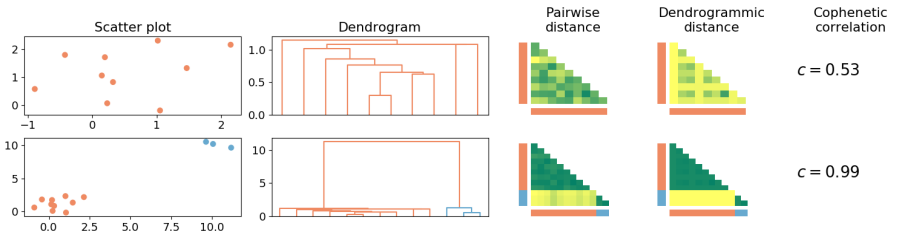


Figure 2.5: Upper: 10 data points form one cluster. Single linkage hierarchical cluster generates the corresponding dendrogram. There is a clear difference between pairwise and dendrogrammic distances, shown in triangular matrices with a green (low) to yellow (high) color scale. This difference leads to a low correlation.
Lower: a second cluster (blue) of 3 additional data points. In this situation, the difference in pairwise and dendrogrammic distances is relatively small. This leads to a higher correlation, indicating the presence of multiple clusters.

this situation, the larger distances between clusters dominate the correlation, which is now close to 1.

Algorithm 1 Hierarchical cluster partitioning

```
procedure PARTITION(dendrogram, thr_cc)
    cc = cc(dendrogram) (Equation 2.5)
    if cc > thr_cc then
        d_1, d_2 = cut_at_highest_level(dendrogram)
        return Partition(d_1, thr_cc), Partition(d_2, thr_cc)
    else
        return d
```

2.2.3 Building block 3: Anomaly detection

We propose two approaches for assigning anomaly scores to machines:

Cluster-based this exploits the structure of the clustering that was found. This approach uses the cluster size as an indicator of the machine's health condition. For example, it assumes that small clusters represent faulty machines.

Similarity-based this exploits dissimilarities of machine signatures compared to the healthy machines detected by the similarity metric. This approach

uses the magnitude of a machine's deviation from the majority of the fleet as an indicator of its health condition. For example, it assumes that a faulty machine's signature deviates more compared to the natural deviations within the healthy proportion of the fleet.

Cluster-based anomaly score

The cluster-based anomaly score uses the discovered clustering to assign an anomaly score to each machine. Machines with similar behavior are expected to be in the same cluster. The larger a cluster, the more likely it is to indicate healthy behavior. Each machine's cluster-based anomaly score is simply the fraction of total machines that are not assigned to its cluster, hence an anomaly score close to one indicates a machine that is likely to be faulty. Thresholding this score to classify each machine as either healthy or faulty requires reasoning about the size of the fleet and the expected proportion of faulty machines.

The clustering's cophenetic correlation threshold thr_{cc} controls the cluster partitioning and is the most important parameter for this anomaly score. The parameter can be set depending on the cost of not identifying a faulty machine (a false negative, missed detection) or incorrectly labeling a healthy machine as faulty (a false positive, false alarm). If thr_{cc} is set too high, a faulty machine might be partitioned together with machines considered to be healthy. The anomaly detection procedure would then report faulty machines as being healthy. Multiple small partitions containing machines considered to be healthy would be formed if thr_{cc} is set too low. The cluster-based anomaly score will then report these machines as being faulty. The value of this threshold is set depending on the application.

Similarity-based anomaly score

The pairwise comparisons obtained by the first block offer detailed insights into the deviation of a machine's signature compared to other machines in the fleet. In similar operating conditions, all healthy machines are expected to have similar signatures and low pairwise distances. However, the different signature of a faulty machine would result in a high pairwise distance to healthy machines.

To assess the health status of a machine X, we compare this machine with a representative healthy machine. Assuming healthy machines cover the majority of the fleet, we ensure this comparison by selecting the median of X's pairwise distances. Using outlier-sensitive metrics such as average or extrema would

cause outliers to influence this comparison. Next, this value is normalized by the median distance within the largest cluster.

Formally, the anomaly score of a machine X in fleet F is defined in Equation 2.6, with P the set of all machine pairs within the largest cluster and $s(X, Y)$ the distance of a machine pair (X, Y) (block 1, machine similarity):

$$score(X) = \frac{median_{Y_i \in F \setminus X}[s(X, Y_i)]}{median_{(Y1_i, Y2_i) \in P}[s(Y1_i, Y2_i)]}. \tag{2.6}$$

Rescaling Typically, an anomaly score is defined as a real value between 0 and 1, with 0 indicating completely normal behavior and 1 completely abnormal behavior. However, our proposed anomaly score can be larger than 1 as the median distance of a faulty machine can be larger than the normalization factor of the median distance within the largest cluster. Therefore, we use an exponential squashing function $score_{rescaled}(x)$ to rescale this anomaly score, as defined in Equation 2.7 [83].

$$score_{rescaled,\alpha}(x) = 2^{-\frac{score(x)^2}{\alpha^2}}. \tag{2.7}$$

In this equation, α corresponds to the value resulting in an anomaly score of 0.5: $score_{rescaled,\alpha}(\alpha) = 0.5$. In other words, a hypothetical machine obtains an anomaly score of 0.5 when deviating to the fleet α times the median machine dissimilarity within the healthy cluster. The higher this parameter, the more dissimilar a machine's signature should be before resulting in a score of 1. However, lower values cause higher anomaly scores for minor dissimilarities. In general, the optimal value of α depends on the exact use case and can be used to fine-tune the sensitivity of the fleet-based condition monitoring framework.

2.2.4 Building block 4: Visualization

Visualizing the results of each block can yield insights into the machine health conditions and the framework's predictions. A simple visualization shows results of each block (for example Figure 2.6). In Chapter 3, we show more examples in Figures 3.8, 3.9, and 3.12.

First, both parts of the machine comparison block can give detailed insights. The template has space for preprocessed machine data, to give the user a more detailed view (Figure 2.6a.). A dissimilarity matrix (Figure 2.6b.) shows the result of pairwise machine comparisons. A color map helps to see relative differences, as absolute values depend on the selected comparison method.

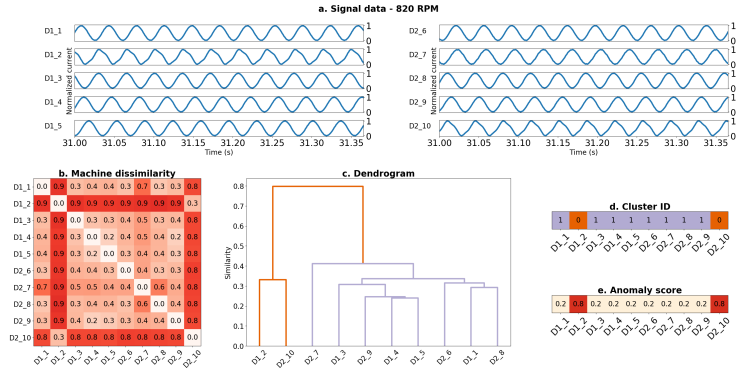


Figure 2.6: An example of the visualizations provided by the framework, which are analyzed in detail in Chapter 3. In this example, drivetrains D1_2 and D2_10 are faulty. Signal data (a.) gives the user a detailed view. In this example, min-max normalization has been applied to the waveforms. A global overview of the fleet is given by pairwise comparisons (b.) and a dendrogram (c.). Finally, the cluster partitions (d.), obtained with $thr_{cc} = 0.9$, and their corresponding anomaly scores (e.) indicate the prediction for a machine's health condition. In this example, the framework correctly identifies D1_2 and D2_10 as being anomalous.

Second, clustering gives a more global overview of the machine fleet. A dendrogram can, for example, show a cluster hierarchy, to detect divergent machines (Figure 2.6c.). Moreover, colors can indicate cluster partitions (Figure 2.6d.). This helps to tune the threshold thr_{cc}. Finally, the anomaly scores correspond to the predicted health conditions (Figure 2.6e.).

2.3 Experiments

To evaluate the framework's applicability for fleet-based health monitoring, we pose four research questions:

1. Can we correctly identify faulty machines in a fleet of similar operating machines?
2. What are the main differences between the cluster-based and similarity-based anomaly scores?
3. Is the framework capable of detecting early (gradual) degradation?

4. How does the size of the fleet relate to the framework's performance?

To answer these questions, we study the outcome of three experiments using carefully collected artificial data sets. This allows studying various situations of interest in a controlled setting.

2.3.1 Experiment 1: A fleet with malfunctioning machines

This experiment studies research questions 1 and 2 by analyzing and comparing the performance of the cluster-based and similarity-based anomaly detection. For evaluation purposes, we employ a simulated machine fleet data set.

Dataset We simulate a data set a fleet of 20 machines, consisting of 17 healthy machines, one having a minor fault (ID 16) and two having a severe but similar issue (ID $19 - 20$). A machine i's signature is represented by a two-dimensional data point $(D1_i, D2_i)$. The values of $D1$ and $D2$ are randomly drawn from a Gaussian distribution $R \sim \mathcal{N}(0, 0.05)$. Exceptions are the $D2$ values of the faulty machines, being set to $D2_{16} = 1,25; D2_{19} = D2_{20} = 2,5$.

Evaluation In this experiment, the distance $s(X, Y)$ of a machine pair (X, Y) is calculated by their data point's Euclidean distance. Next, these distances are used to cluster the fleet through hierarchical clustering with *complete* linkage. Finally, the hierarchical cluster is partitioned with thr_{cc} set to 0.9. Figure 2.7b visualizes this clustering in a dendrogram. The cluster partitioning results in three cluster partitions: a majority cluster of size 17 containing the healthy machines, a cluster of size 1 with machine 16, and a cluster of size 2 having machines 19 and 20.

The cluster-based anomaly scores, shown in Figure 2.7c, clearly do not represent a machine's health status. The score of machine 16 is higher than those of machines 19 & 20, while the former is only affected by a minor fault. The cluster-based anomaly score assumes a fault is more severe if it affects fewer machines, which is invalid in many cases.

We obtain the similarity-based anomaly scores in a two-step approach. First, we calculate the score according to Equation 2.6. In this case, machines 1 – 15, 17, and 18 form the largest cluster P. The anomaly score of any machine X is thus the median distance from X to the aforementioned machines, divided by the median distance between any machine pair in this largest cluster P. Second, we rescale the obtained valued according to Equation 2.7, using $\alpha = 3$. This results in an anomaly score of 0.5 for a hypothetical machine, deviating

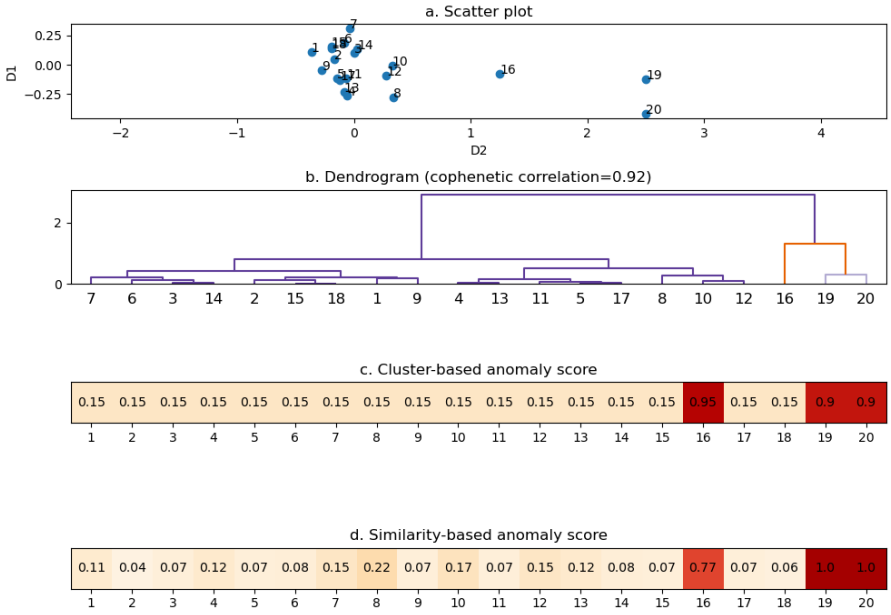


Figure 2.7: The cluster-based anomaly score assumes a fault is more severe if fewer machines are affected by it, which is invalid in many cases. Differences in signatures compared to healthy machines, used by the similarity-based approach, are a better health indicator.

to the fleet of three times the median dissimilarity within the healthy cluster. The resulting anomaly scores, shown in Figure 2.7d, are representative of the machines' health statuses: minor faulty machine 16 is given a lower anomaly score compared to machines 19 and 20.

Using the similarity-based anomaly score thus helps a user interested in the true machine health status, as this score is related to the dissimilarity of a machine to the healthy proportion of the fleet. In many cases, it is safe to assume that a higher anomaly score should indicate a more severe machine fault. Additionally, visualizing the framework's blocks helps the end-user to better understand the health status prediction. For example, the anomaly scores indicate a severe fault for machines 19 and 20, while the dendrogram also suggests their signatures are similar. An operator could thus infer that both machines suffer a similar fault.

2.3.2 Experiment 2: A fleet of gradual degrading machines

Machine wear can cause both sudden failure and gradual degradation where a machine's performance decreases (slowly) over time. The maintenance related to this gradual degradation is typically less urgent compared to repairing sudden failures. Within a machine fleet, the various machines can be exhibiting different levels of degradation. This experiment evaluates the performance of the cluster-based and similarity-based anomaly scores in such a fleet. More specifically, we study if our framework can detect early degradation (research question 3) and if the anomaly scores indicate the severity of the fault (research question 2)

Dataset In this experiment, we simulate a fleet of machines exhibiting different levels of degradation. This data set is similar to the data set of Section 2.3.1. We consider a fleet of 20 machines but simulate gradual degradation by perturbating the signatures of machines $15 - 20$. For these machines, we alter $D2$ in their two-dimensional signature $(D1_i, D2_i)$ as shown in Table 2.1 and Figure 2.8a.

Table 2.1: We simulate machine degradation by altering the signatures $(D1_i, D2_i)$ of machines $15 - -20$ by perturbating $D2$.

ID	15	16	17	18	19	20
$D2$	0.5	1	1.5	2	2.5	2.5

Evaluation We perform complete link hierarchical clustering using Euclidean distance and partition the obtained clusterings using $thr_{cc} = 0.9$, as shown in Figure 2.8b.

While some separation between machines 1-14 and 15-20 exists, it is not substantial enough to detect multiple clusters. Hence, this results in an identical cluster-based anomaly score of 0 for each machine, as shown in Figure 2.8c. These scores do not allow an operator to differentiate between machine 1, supposedly in good health condition, and machine 19, which is clearly the most deviating. In contrast, the individualized similarity-based anomaly scores shown in Figure 2.8d show distinct values for each machine. Moreover, they show clear increased anomaly scores for the machines that are degrading, which yields a more nuanced view of each machine's health status.

Similar to the previous experiment, the similarity-based anomaly scores are calculated using Equation 2.6. In this example, the largest cluster P contains all machines, including those that are faulty. Our anomaly score is robust to this since it makes use of the median distances. As only a minority of the machines

are faulty, using the median distance always results in a comparison with a healthy machine. Finally, we rescale the anomaly score by Equation 2.7 with $\alpha = 3$.

We conclude that similarity-based anomaly detection offers three benefits relevant for condition monitoring compared to cluster-based anomaly detection. First, similarity-based anomaly detection can reliably detect (early) machine degradation. Cluster-based anomaly detection is highly dependent on the health status of the machines in the fleet. For example, clustering might be unable to isolate machines in a fleet where the ages of the machines vary greatly, which leads to a high variance in the amount of degradation exhibited by the individual machines. Second, only similarity-based anomaly scores are related to the severity of a machine fault. In many cases, this severity corresponds to the dissimilarity of the machine's signature compared to the signatures exhibited by the majority of the fleet. In contrast, cluster-based anomaly scores only indicate the number of machines with a similar health status. Third, similarity-based anomaly scores assess each machine individually. This provides more fine-grained insights into the fleet's health status compared to the group-level insights provided by the cluster-based anomaly scores.

2.3.3 Experiment 3: A fleet with a limited number of machines

The framework requires the majority of the units to be healthy to be able to detect deviating signatures. However, the minimal number of healthy units required to obtain good performance varies. In this experiment, we study the factors that influence this minimum number of machines, answering research question 4. These insights help a user to decide on the applicability of the health monitoring framework, given a specific use case.

We use an artificial example to easily manipulate the data set's properties, such as the severity of the faults and the number of units. This allows to thoroughly study the effect of these parameters on the minimal number of healthy machines required by the framework to result in good prediction performance.

Data set This experiment makes use of an artificial data set, containing n_H healthy and n_F faulty units represented by a two-dimensional data point. Each unit is sampled from a Gaussian distribution, corresponding to its health state. We use the standard normal distribution ($\mathcal{N}(\mu = 0; \sigma^2 = 1)$) to generate the healthy units, while sampling the faulty units from a distribution $\mathcal{N}(\mu = \alpha; \sigma^2 = 0.16)$. The parameter α defines the distance between these distributions' means,

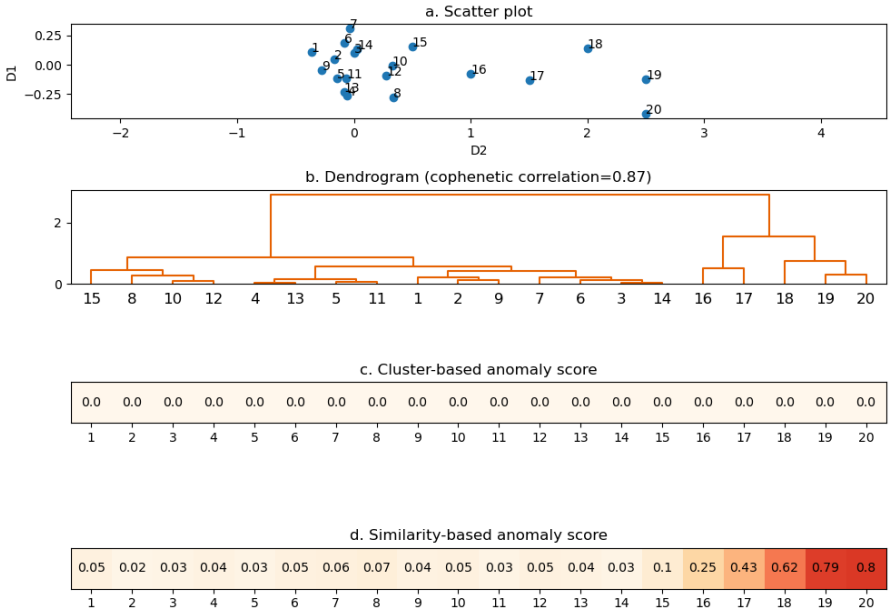


Figure 2.8: The similarity-based anomaly score assesses each machine's health status individually and detects gradual degradation. This is not possible using the cluster-based anomaly score, as all machines within a cluster receive the same score.

representing the severity of the faults. The higher this parameter is, the larger the distance between both groups and the more severe the simulated fault is.

Evaluation We evaluate different data sets, varying both the severity of the fault α and the number of faulty units n_F. For each data set, we determine the minimum required number of healthy units $n_{H_{min}}$ to obtain an F1-score of at least 0.9. Clusters are partitioned with $thr_{cc} = 0.9$ and we consider units with a similarity-based anomaly score of $score_{rescale,1.5}(X) \geq 0.5$ as faulty. For statistical reliability, we repeat this experiment 20 times and report the average values.

The results shown in Table 2.2 indicate that detecting severe faults only requires a minimal majority of $n_F + 1$ healthy units. The underlying distributions are clearly distinct and rough estimates based on the limited data is sufficient for fault detection purposes. However, the difference between the healthy and the

slightly impacted units is lower. In such case, the framework requires a better estimate (and thus more data) of the healthy units' distribution.

Table 2.2: Minimal required number of healthy units $n_{H_{min}}$, for various severities of the fault α and number of faulty units n_F. Non-integer values are caused by averaging multiple test for statistical reliability.

		α								
		2.0	**2.5**	**3.0**	**3.5**	**4.0**	**5.0**	**6.0**	**8.0**	**10.0**
n_F	**1**	3.38	2.32	2.15	2.10	2.05	2.00	2.00	2.00	2.00
	2	11.08	6.94	4.21	3.80	3.45	3.00	3.00	3.00	3.00
	3	25.50	11.56	8.21	5.63	4.95	4.35	4.00	4.00	4.00
	4	31.25	18.00	12.33	9.32	6.47	5.40	5.00	5.00	5.00
	5	35.00	26.87	15.80	11.25	9.10	6.45	6.00	6.00	6.00

2.3.4 Discussion

The experiments help to answer our four research questions regarding the framework' applicability to fleet-based health monitoring:

Can we correctly identify faulty machines in a fleet of similar operating machines?

Our framework can detect faulty machines in a simulated machine fleet. However, our experiments show that the framework's configuration can drastically impact the framework's performance. Using domain knowledge about the application of interest can help a user select the most appropriate configuration for our framework.

What are the main differences between the cluster-based and similarity-based anomaly scores?

We conclude that the similarity-based anomaly score offers multiple benefits compared to the cluster-based anomaly score:

1. Similarity-based anomaly detect assesses each machine individually. This enables more fine-grained insights compared to the cluster-level scores provided by the cluster-based anomaly detection.

2. Only similarity-based anomaly scores correspond with the severity of the machine fault. The cluster-based anomaly scores purely represent the proportion of machines in a similar state.

3. Similarity-based anomaly scores are obtained independently of the clustering. This is a clear benefit if the clustering algorithm fails to isolate faulty machines, for example, due to a fleet with machines exhibiting different levels of degradation.

Is the framework capable of detecting early (gradual) degradation?

The framework can detect gradual degrading machines when using the similarity-based anomaly scores. This helps end-users by allowing them to monitor the fleet and determine at any early stage when preventive maintenance should be scheduled.

How does the size of the fleet relate to the framework's performance?

Experiment 3 indicates that the size of the fleet impacts the framework's sensitivity, i.e., how much a faulty machine's signature must differ from the healthy machines' signatures in order for our framework to correctly identify the faulty machine. Detecting a drastically different machine signature is relatively easy and does not require a large fleet. However, detecting minor deviations is more challenging and requires a large number of healthy machines. If the severity of a fault impacts the magnitude of the deviation in the machine's signature, this implies that minor faults can only be detected in large fleets.

2.4 Conclusion

This chapter proposes a novel machine fleet condition monitoring framework that offers several advantages over both traditional condition monitoring approaches and supervised machine learning techniques. First, it removes the need to know specific potential machine faults a priori. Instead, fleet-based health monitoring can detect deviations in the machines' signatures based on generic similarity metrics. Second, by making the comparison between pairs of simultaneously operating machines, fleet monitoring does not require a high-quality historical (labeled) data set for training. Both traditional signal processing approaches and supervised machine learning need such data to learn to recognize the characteristics of a faulty machine's exhibited behavior. Moreover, these require

ground-truth knowledge about the machine's actual health status. This is not required for fleet monitoring, as it assumes the majority of the machines to be healthy. Third, fleet monitoring allows analysis in dynamic operational conditions. Online comparisons allow detecting deviating machine behavior even in unconsidered operational conditions. Finally, the framework offers a high level of interpretability. Visualizations allow a domain expert to get insights into the predictions and gain confidence in the methodology. This is especially in contrast with black-box machine learning techniques, whose models are very hard to interpret.

Our experiments show that for simulated machine fleets, the framework correctly identifies faulty machines in a fleet. By using similarity-based anomaly detection, we can quantify how much a machine deviates from its expected behavior. In many cases, the magnitude of this deviation indicates the severity of the fault. Finally, we show that the severity at which a machine failure is detected depends on the proportion of healthy units in the fleet. The larger this proportion is, the easier it is to detect small deviations.

Chapter 3

Applications of fleet-based anomaly detection

This chapter is based on:

- K. Hendrickx, W. Meert, Y. Mollet, J. Gyselinck, B. Cornelis, K. Gryllias, and J. Davis. A general anomaly detection framework for fleet-based condition monitoring of machines. *Mechanical Systems and Signal Processing*, 139:106585, may 2020. ISSN 08883270. doi: 10.1016/j.ymssp.2019.106585
- K. Hendrickx, W. Meert, B. Cornelis, K. Gryllias, and J. Davis. Similarity-based anomaly score for fleet-based condition monitoring. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society, PHM*. PHM Society, 2020
- K. Hendrickx, W. Meert, J. P. D. C. Patricio, B. Cornelis, K. Gryllias, and J. Davis. Fleet-based health monitoring for end-of-production-line and operational testing. In *Online Proceedings of ISMA2020*, Leuven, Belgium, 2020. KU Leuven

3.1 Introduction

This chapter studies the applicability of the proposed fleet-based condition monitoring framework in operational monitoring and end-of-production-line testing applications.

Operational monitoring We study two operational monitoring applications. The first application analyzes electrical signature data, while the second application focuses on vibration analysis. Both involve a fleet of electrical drivetrains where all machines in the fleet are simultaneously operated in comparable operational conditions. In both applications, the objective is to detect a voltage unbalance fault. If undetected, such a fault results in irreversible wear due to increased temperature and additional vibrations which ultimately lead to premature machine failure. In practice, this fault is typically caused by a grid fault or a bad wire connection.

The first application illustrates how various ways of instantiating our framework affect its performance. Each variant uses different signal domains and machine comparison metrics, to demonstrate the framework's broad applicability and capacity to incorporate different degrees of domain knowledge.

End-of-production-line testing This application assesses the quality and integrity of newly-produced gears, which are tested at the end of their production process. Because our framework was originally designed to perform simultaneous online comparisons, we must adapt it to handle this sequential setting. We do so by considering a short-term historical database consisting of the most recently-tested gears.

We study the framework's performance through three performance metrics: Precision, Recall, and F1-score. These consider the two types of misclassifications: false negatives (FN) and false positive (FP) (Table 3.1). Precision (Equation 3.1a) and Recall (Equation 3.1b) help the user to take the cost into account of labeling a healthy machine as being faulty (precision, false alarm) versus missing a faulty machine (recall, missed detection). The F1-score (Equation 3.1c) considers both faults equally important and is used as a single performance measure.

$$Precision = \frac{tp}{tp + fp} \tag{3.1a}$$

$$Recall = \frac{tp}{tp + fn} \tag{3.1b}$$

Table 3.1: Possible prediction mistakes

		True condition	
		Faulty	Healthy
Predicted condition	Faulty	True positive tp	False positive fp
	Healthy	False negative fn	True negative tn

$$F1 - score = 2\frac{Precision.Recall}{Precision + Recall} \tag{3.1c}$$

3.2 Application 1: Machine health monitoring through electrical signature analysis

This application involves electrical measurements of an electrical machine fleet. The task is to detect a voltage unbalance fault based on current signals. Three variants show the general applicability of the framework in different signal domains. The first variant uses time-domain data and demonstrates a generic approach for waveform analysis. The second variant is an example of how to include domain knowledge. Finally, the third variant is a hybrid method and analyzes time/frequency data.

Our experiments consider stationary as well as dynamic operational conditions. The former allows analyzing the consistency of the framework, the latter its generality. Within these operational conditions, the health status of machines D1_2 and D2_10 is varied.

3.2.1 Experimental fleet test rig and data set

A single-phase current of ten electrical drivetrains (Figure 3.1) is measured at 25600 Hz. Each drivetrain consists of an electrical motor pair, connected by a flexible jaw coupling. A 3-phase Squirrel cage induction motor (SCIM)[1] drives the shaft. Ten ABB[2] drive controllers manage the drive sides with internal closed-loop direct-torque control. All drivetrains are set up to always have similar speeds. The other motor acts as load and is either a 3 kW Direct Current (DC) motor (drivetrains D1_1 – D1_5) or a 3 kW Wound Rotor

[1] Siemens 1AV3105B 1LE10031AB522AA4
[2] ABB ACS880-01-09A4-3

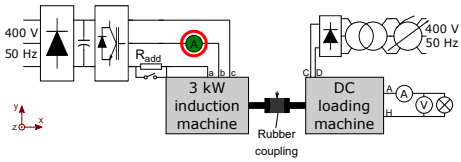
Synchronous Motor (WRSM) (drivetrains D2_6 – D2_10). The specifications of each machine are shown in Table 3.2. Resistors generate load while keeping the current excitation constant. The resulting load torque is proportional to speed and corresponds to the rated load at rated speed.

A 3 Ω external resistor R_{add} emulates a voltage unbalance in drivetrains D1_2 and D2_10. This resistor is inserted between the drive controller and the motor (Figure 3.1b). A higher resistance would trigger the internal safeties of the ABB drive controller.

The data set includes both stationary and dynamic system behavior under loaded and unloaded operating conditions. For the stationary case, each drivetrain is running at a fixed speed (820 RPM or 1500 RPM). On the other hand, during the dynamic operational condition, all machines perform a run-up from 0 RPM to 1200 RPM. In our experiments, we analyze and predict the machine status for 0.5s non-overlapping analysis windows. A machine is considered to be faulty after being identified as anomalous for five consecutive windows. This reduces misclassifications due to signal noise.



(a) Motor fleet.



(b) Schematic of drivetrain D1_2. The analyzed current sensor (green) measures phase B.

Figure 3.1: The measurement setup consists of ten drivetrains, having a 3kW SCIM (drive, inner part of the fleet) coupled to a 3kW WRSM (left) or DC (right) motor (load). Drive and load side motors are connected with a rubber coupling.

3.2.2 Classic voltage unbalance diagnosis techniques

Motor Current Signature Analysis (MCSA) is a popular class of techniques for detecting motor faults that apply signal processing methods to current signals. It can be used to detect a voltage unbalance [105, 106]. Some of these methods require measuring multiple motor current signals. For example, the phase shift between the three current signals of a healthy system is 120 degrees. A voltage unbalance results in changes to this angle [107]. Another example measures an

Table 3.2: Rated parameters of the motors

	Power	Phase-to-phase voltage	Current	Frequency	Pole pairs	Speed
SCIM	3 kW	400 V	6.2 A	50 Hz	2	1385 RPM
WRSM	3 kW	400 V	4.3 A	50 Hz	2	1500 RPM
DC	3 kW	270 V	11.2 A	DC	1	1500 RPM

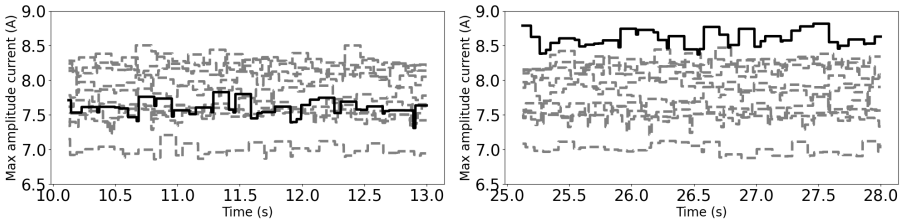
estimator for I_{RMS} based on two[3] current signals. A fluctuating component at twice the fundamental frequency, the second harmonic, is observed due to the presence of inverse sequence current in the stator [106]. Formally, the i-th harmonic is defined by Formula 3.2 with $fundamental(X)$ being the frequency of the lowest periodic waveform in signal X.

$$harmonic_i(X) = i * fundamental(X) \tag{3.2}$$

A single current sensor is sensitive to a secondary effect of the voltage unbalance [106]. The oscillations of the shaft lead to an increase in the third harmonic. This work compares the framework with three approaches that make use of this signal. First, careful time-domain analysis reveals an increase in the current amplitude. However, a fleet-based approach cannot use this current amplitude as the difference due to a fault is lower than the variability within the fleet (Figure 3.2). Moreover, the current amplitude highly depends on the operational conditions (i.e. rotational speed and load). Second, the fault affects the current waveform. However, the shape of the waveform depends on the operational speed (Figure 3.3). Any type of deviation should thus trigger a machine fault prediction. Finally, the frequency domain shows an increase of the third harmonic amplitude (Figure 3.4). The harmonic amplitude is not sensitive to changes in speed but does depend on the load. Different load conditions could thus require different thresholds.

In each of these approaches, the machine is affected up to a certain speed ($\sim$ 1400RPM). This is demonstrated in Figure 3.4, where the amplitude of the third harmonic decreases at that point. The hypothesis is that at higher speeds, a flux weakening region is entered. For example at the rated load above

[3]The third current signal can be inferred from the measured signals.



(a) All machines healthy (b) D2_10 (orange) faulty

Figure 3.2: Voltage unbalance detection based on amplitude is not possible in a fleet context. Machine D2_10 is indicated in black. The variance within the healthy fleet is larger than its amplitude increase due to a voltage unbalance. All machines are running stationary at 820 RPM (loaded).

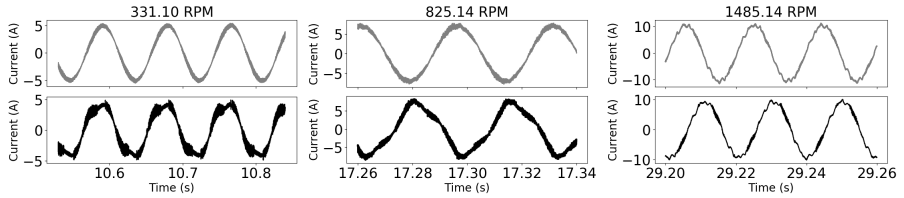


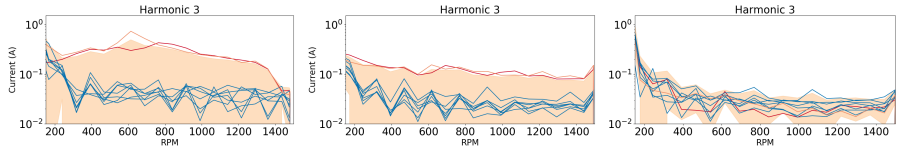
Figure 3.3: Current waveform analysis can detect a voltage unbalance. The fault affects the current waveform of a faulty drivetrain (black). The exact shape is however dependent on the operational speed. Above 1385 RPM, the fault effect disappears due to an increase of the flux. A current signal of a healthy drivetrain (gray) is shown for reference.

1385 RPM, the stator frequency exceeds the rated 50 Hz. Therefore, the flux is decreased in the machine to keep the stator voltage under its limit. As a consequence, the magnetic saturation level decreases, and the third harmonic in the current becomes less visible.

The remainder of this use case utilizes a simple condition monitoring method as a benchmark to validate the proposed framework. This method considers a machine as healthy when its third harmonic amplitude is within σ standard deviations of the fleet's mean. Note that this is a simplified fleet approach, which avoids setting thresholds manually. Table 3.3 shows the fault detection performance for this methodology with different values of σ, Figure 3.4 illustrates the specific case for $\sigma = 2$.

Table 3.3: Performance of classic voltage unbalance detection using the third current harmonic. Scores corresponding to a scenario's highest F1-score are indicated in bold.

		σ							
	Metric	0.5	1	1.5	2	2.5	3	3.5	4
820 RPM load	Precision	0.174	0.236	0.289	0.323	**0.334**	0.334	0.334	0.334
	Recall	0.857	0.857	0.845	0.793	**0.758**	0.695	0.641	0.596
	F1	0.289	0.371	0.431	0.459	**0.464**	0.451	0.440	0.428
820 RPM no load	Precision	0.157	0.213	0.259	0.285	**0.297**	0.297	0.297	0.297
	Recall	0.855	0.855	0.855	0.800	**0.770**	0.706	0.652	0.605
	F1	0.265	0.341	0.397	0.421	**0.429**	0.418	0.408	0.399
1500 RPM load	Precision	0.145	**0.164**	0.172	0.172	0.172	0.172	0.172	0.172
	Recall	0.623	**0.523**	0.422	0.337	0.307	0.281	0.259	0.241
	F1	0.235	**0.250**	0.244	0.228	0.220	0.213	0.207	0.201
1500 RPM no load	Precision	0.142	0.190	0.236	0.273	**0.283**	0.283	0.283	0.283
	Recall	0.855	0.855	0.855	0.832	**0.796**	0.730	0.673	0.625
	F1	0.243	0.311	0.370	0.411	**0.418**	0.408	0.399	0.390
Run up load	Precision	0.245	0.325	0.390	0.444	**0.465**	0.465	0.465	0.465
	Recall	0.833	0.827	0.824	0.822	**0.815**	0.747	0.689	0.640
	F1	0.378	0.467	0.530	0.577	**0.592**	0.573	0.556	0.539
Run up no load	Precision	0.232	0.302	0.361	0.410	**0.427**	0.427	0.427	0.427
	Recall	0.821	0.786	0.768	0.757	**0.737**	0.676	0.624	0.579
	F1	0.361	0.436	0.491	0.532	**0.540**	0.523	0.507	0.491



(a) Load, D1_2 & D2_10 faulty.

(b) No load, D1_2 & D2_10 faulty.

(c) No load, all machines healthy.

Figure 3.4: The third harmonic can be used to detect a voltage unbalance. Faulty machines D1_2 (red) and D2_10 (orange) show a clear increase compared to the healthy (blue) machines between 300 and 1420 RPM. The orange band indicates the range in which a machine is considered healthy, being 2 standard deviations around the mean. This correctly detects the machine health condition in most of the cases (Table 3.3).

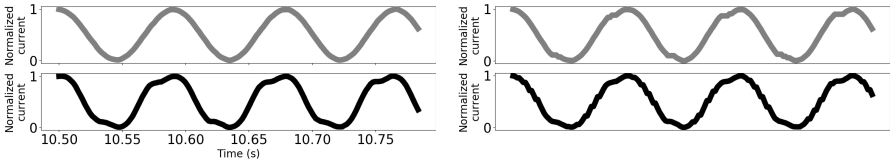
3.2.3 Variant 1: Time domain

This variant evaluates the different operational conditions in the time domain representation. The goal is to detect any deviation on the current waveform with a generic comparison measure. Additionally, we also assess the two cluster-based and similarity-based anomaly scores' ability to observe gradual changing machine behavior. More specifically, we compare how well the anomaly scores represent a machine's health status while facing varying severeness of degradation.

Block 1: Machine comparison

Since the differences appearing in current amplitudes cannot be used in a fleet-based approach (Figure 3.2), min/max normalization is performed. Moreover, the current signals are downsampled to 50 samples per period. This removes the effect of signal noise and allows the analysis to focus on the waveforms.

The waveform of a machine affected by a voltage unbalance depends on the operational speed. Technique that assess this shape are thus required to be robust to changes due to varying operational settings. Dynamic Time Warping (DTW) is a popular technique for pattern recognition, but the classic implementation considers waveforms of healthy and faulty machines as similar (Figure 3.5). However, aligning healthy and faulty signals involves significant amounts of warping. In contrast, little warping occurs when comparing two healthy machines. Therefore, the amount of warping (Equation 2.4) can be used as a powerful indicator for waveform deviations, regardless of the exact shape. The Ψ-DTW extension avoids unnecessary warping in the boundary



(a) Measured current signals.

(b) Warped current signals. The time axis is hidden, as the two signals have been non-linearly warped

Figure 3.5: DTW of a healthy (gray) and faulty (black) machine. While the measured current signals have a clear difference in shape (a), DTW can non-linearly align them so the signals become similar (b).

points. Unaligned start- and endpoints would otherwise dominate the DTW warping amount measure. The used value of Ψ is corresponding to half a period of the current signal, allowing to align any phase offset.

Block 2: Fleet clustering

Hierarchical clustering, combined with a cophenetic correlation partition procedure, generates machine clusters. First, hierarchical clustering uses pairwise machine comparisons and single linkage to construct a cluster hierarchy. Single linkage considers the minimum distance as the similarity of two subclusters. The cophenetic correlation procedure partitions this hierarchy with recursive top-down evaluations, if the cophenetic correlation is larger than thr_{cc} (Algorithm 1).

The value of thr_{cc} affects the effectiveness and the performance of the clustering but its optimal value is application dependent. Therefore, the framework is evaluated for various values of thr_{cc} to show the parameter's impact on predictive performance. A lower value results in more clusters, which leads to more machines incorrectly labeled as faulty (low precision, higher false alarm rate). A higher value, on the other hand, leads to more misses of faulty machines (low recall, higher missed detection rate).

Block 3: Anomaly detection

The cluster-based anomaly score is defined as the fraction of machines outside each cluster. This is a simple scoring mechanism to demonstrate the concept of

anomaly detection. The anomaly threshold $thr_{ad} = 2/3$: if a machine's score is above this threshold, it is predicted to be faulty.

The similarity-based anomaly score (Equation 2.6) combines the output of the machine comparison and clustering blocks. It indicates how much a machine deviates from the fleet relative to the median deviation within the healthy cluster. The anomaly scores are re-scaled according to Equation 2.7 with $\alpha = 1.4$. Using this equation and value of α converts an unscaled score of 1.4, into a scaled anomaly score of 0.5 (Equation 3.3).

$$score_{rescaled,\alpha=1.4}(x) = 2^{-\frac{score(x)^2}{1.4^2}}. \tag{3.3}$$

Block 4: Visualization

Visualizations allow a domain expert to interpret the framework's predictions. By showing the results of each block, an expert can confirm the predictions and avoid taking unneeded actions. All variants of this application use the default visualization of Section 2.2.4, being a (normalized) signal domain representation, machine dissimilarity, clustering, and anomaly scoring.

Experiment 1: Cluster-based vs similarity-based anomaly detection

The objective of this experiment is to analyze the differences between the cluster-based and similarity-based anomaly scores in a real-world health monitoring application.

We analyze a subset of the fleet test rig data set. Specifically, we consider dynamic operational conditions where the drive trains perform a run-up from 500 to 1500 RPM. The increasing operational speed causes the severity of the voltage unbalance to gradually decrease due to the flux weakening discussed in Section 3.2.2. Such a run-up is thus an ideal use case to study the framework's ability to identify gradual degradation in machines.

Figures 3.6 and 3.7 show the relation between each drive train's speed and anomaly score for the cluster-based and similarity-based methods respectively. Both detect the voltage unbalance fault up until around 1250 RPM. However, the cluster-based score remains constant as long the fault can accurately be detected. When the voltage unbalance becomes less prominent, the score abruptly drops to 0. Moreover, minor fluctuations in the signal cause the anomaly score of healthy machines to spike at various speeds. Based on these scores, one could suspect a severe health issue for all machines. In contrast, the

similarity-based score gradually decreases from 1 to 0.4 during the same period, aligning with the decreasing severity of the fault. Moreover, the variation of the healthy machines' anomaly scores is limited and offers better insights into the fleet's health conditions. For example, at low speed, one could observe a slight deviation for one particular machine. While we did not intentionally introduce a fault for this machine, the similarity-based score suggests it is slightly degraded. However, we are unable to confirm this hypothesis.

We conclude that the cluster-based anomaly score only allows a hard health status prediction, as it considers a machine as either healthy or faulty. An anomaly score based on similarities instead of cluster sizes offers several benefits. First, the number of machines in each cluster is independent of how deviating (and thus faulty) a machine is. For example, the cluster-based score decreases if more machines face the same fault. Second, when machine transitions to another cluster, its cluster-based score can abruptly and drastically change. Using similarities is particularly beneficial for edge cases that would transition due to only a slight change in the machine's health status. In such a case, the similarity-based anomaly score changes more gradually. Third, the similarity-based anomaly score can be calculated on a machine level. This individualized score provides a nuanced prediction and is beneficial to detect early degradation, allowing to schedule predictive maintenance. For the cluster-based anomaly score, slight deviations would remain hidden until severe enough to transition the machine outside the majority cluster.

Experiment 2: Performance of fleet-based condition monitoring

This experiment compares our fleet-based condition monitoring with the traditional condition monitoring approach discussed in Section 3.2.2. The objective is to validate our hypothesis that our approach outperforms such traditional analysis.

The example of Figure 2.6 shows the evaluation of a single analysis window. A domain expert can use this visualization to validate the predictions or to tune the general framework. The time series show deviating behavior of machines D1_2 and D2_10 in each of the subfigures corresponding to the different building blocks. We identify a machine as faulty after being identified as anomalous ($score_{rescaled,1.4}(X) \geq 0.5$) for five consecutive windows.

We use the performance metrics presented in Table 3.1 to evaluate the framework's ability to detect a deviating waveform. Both the benchmark (Table 3.3) and the similarity-based anomaly detection approaches of our framework (Table 3.4) are able to detect an unbalance, except at 1500 RPM in a loaded condition. In this condition, the decrease of the flux hides the effects

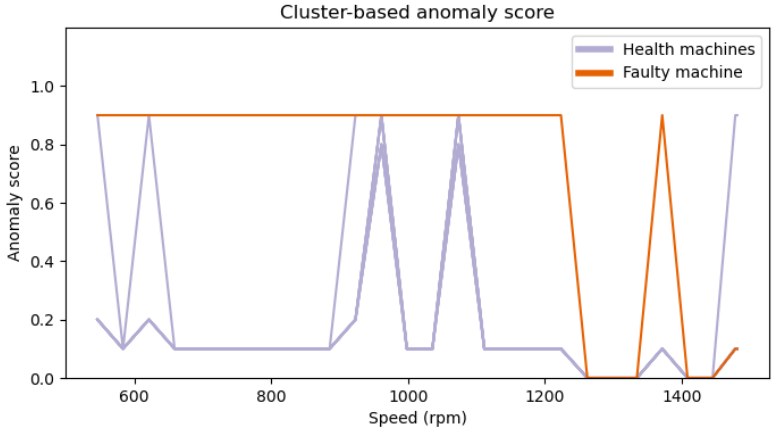


Figure 3.6: The cluster-based anomaly score has very abrupt transitions. Moreover, minor signal fluctuations cause spikes in the anomaly scores of healthy machines, as a machine is either considered to be healthy or faulty. It is impossible for an operator to assess the severity of a fault.

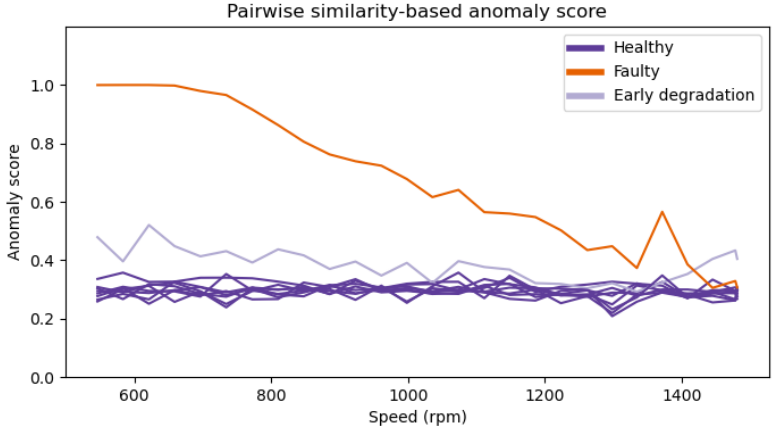


Figure 3.7: The similarity-based anomaly score gradually decreases as speed increases, following the decreasing severity of the fault. Additionally, we observe variation within the fleet of healthy machines, with the anomaly score of one machine being consistently higher compared to the other healthy drive trains. We hypothesize that this is due to the early degradation of this drive train.

of the voltage unbalance. In other stationary scenarios and the dynamic run-up scenarios, our framework outperforms the benchmark method. Moreover, the used machine comparison block of our framework offers the advantage that it detects any deviations in the current waveforms, not only caused by an increase of the third harmonic.

Albeit still outperforming the benchmark, the precision is relatively low at 820 RPM in loaded conditions. As demonstrated in Experiment 1, the similarity-based anomaly score better detects early degradation. The framework detects this drive train to be slightly anomalous, which leads to a lower performance due to incorrectly flagging this drive train as a false positive. Indeed, omitting this drive train from the analysis drastically increases the precision (Table 3.5.

Finally, we observe that the cluster partitioning parameter thr_{cc} only minorly impacts the framework's performance. As the performance indicators are consistently high for $thr_{cc} = 1$ (not partitioning the clustering), our results suggest that clustering is not required for the similarity-based anomaly score in a fleet of homogeneous operating machines.

3.2.4 Variant 2: Frequency domain

This variant demonstrates the case in which detailed domain knowledge is included. A voltage unbalance affects the third harmonic of the fundamental frequency, which is used in this variant to compare the machines. In general, this is very similar to the benchmark method, as both depend on domain knowledge.

Block 1: Machine comparison

A Fast Fourier Transformation (FFT) converts time series within an analysis window to their frequency content, which is converted to a log scale and normalized using min/max scaling. No outliers are present in the considered data set, other cases might need more outlier-robust techniques. Machines are represented in the feature space, by the amplitude or their current's third harmonic. This increases if one of the machines is faulty, as shown in Section 3.2.2.

The periodic nature of current signals allows for estimating the fundamental frequency. This corresponds to the highest peak in the frequency spectrum. The third harmonic is found at three times this frequency. The maximum value in a small window of $\pm$ 5Hz around the estimated frequency is used as its amplitude.

Table 3.4: Performance of the proposed framework for fault detection using electrical signature analysis via waveform comparison. Scores corresponding to a scenario's highest F1-score are indicated in bold. For reference, the last column shows the performance of the classic diagnostics approach (Table 3.3) with the highest F1-score. In five out of the six cases, the proposed approach obtains the best F1 score.

		thr_{cc}							Classic approach
	Metric	0.5	0.7	0.8	0.85	0.9	0.95	1	
820 RPM load	Precision	0.203	0.480	0.511	0.516	0.516	0.516	**0.627**	*0.334*
	Recall	0.857	0.857	0.857	0.857	0.857	0.857	**0.857**	*0.758*
	F1	0.329	0.615	0.640	0.644	0.644	0.644	**0.725**	*0.464*
820 RPM no load	Precision	0.195	0.591	0.817	0.895	0.949	0.949	**1.000**	*0.297*
	Recall	0.855	0.855	0.855	0.855	0.855	0.855	**0.855**	*0.770*
	F1	0.318	0.699	0.836	0.874	0.900	0.900	**0.922**	*0.429*
1500 RPM load	Precision	**0.095**	0.024	0.000	0.000	0.000	0.000	0.000	*0.164*
	Recall	**0.318**	0.009	0.000	0.000	0.000	0.000	0.000	*0.523*
	F1	**0.146**	0.013	0.000	0.000	0.000	0.000	0.000	*0.250*
1500 RPM no load	Precision	0.475	0.989	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	*0.283*
	Recall	0.855	0.855	**0.855**	**0.855**	**0.855**	**0.855**	**0.855**	*0.796*
	F1	0.610	0.917	**0.922**	**0.922**	**0.922**	**0.922**	**0.922**	*0.418*
Run up load	Precision	0.329	0.800	0.895	**1.000**	**1.000**	**1.000**	**1.000**	*0.465*
	Recall	0.852	0.741	0.630	**0.630**	**0.630**	**0.630**	**0.630**	*0.815*
	F1	0.474	0.769	0.739	**0.773**	**0.773**	**0.773**	**0.773**	*0.592*
Run up no load	Precision	0.224	0.828	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	*0.427*
	Recall	0.857	0.857	**0.857**	**0.857**	**0.857**	**0.857**	**0.857**	*0.737*
	F1	0.356	0.842	**0.923**	**0.923**	**0.923**	**0.923**	**0.923**	*0.540*

Table 3.5: The framework's performance in operational testing. Omitting the unintentionally degradation in the drive train yields to better results.

		thr_{cc}							Classic diagnostics approach
	Metric	0.5	0.7	0.8	0.85	0.9	0.95	1	
820 RPM load	Precision	0.259	0.932	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	*0.334*
	Recall	0.857	0.857	**0.857**	**0.857**	**0.857**	**0.857**	**0.857**	*0.758*
	F1	0.398	0.893	**0.923**	**0.923**	**0.923**	**0.923**	**0.923**	*0.429*
Run up load	Precision	0.390	**1.000**	1.000	1.000	1.000	1.000	1.000	*0.465*
	Recall	0.852	**0.741**	0.630	0.630	0.630	0.630	0.630	*0.815*
	F1	0.535	**0.851**	0.773	0.773	0.773	0.773	0.773	*0.592*

This variant compares a machine pair by the difference in their third harmonic amplitudes (Equation 3.4).

$$s(X, Y) = |harmonic_3(X) - harmonic_3(Y)| \tag{3.4}$$

Blocks 2 – 4: Fleet clustering, Anomaly detection & Visualization

Clustering, anomaly detection, and visualization make use of the default implementations. Hierarchical clustering combined with cophenetic correlation clusters the fleet. Therefore, the framework is evaluated for various values of thr_{cc} to show the parameter's impact on predictive performance. Anomaly scores use these clusters and are defined as the fraction of machines outside each cluster[4]. This approach considers a score above $^2/_3$ as anomalous behavior. The visualization shows the frequency spectra enriched with the harmonic amplitudes windows.

Results

Figure 3.8 shows the evaluation of a single analysis window. The anomalous behavior of D1_2 and D2_10 is successfully detected.

The evaluation procedure uses performance metrics to evaluate the framework's ability to detect deviating third harmonic amplitudes (Table 3.6). This version of the framework has similar performance compared to the previous variant (Table 3.4). It outperforms the benchmark method (Table 3.3) in almost all stationary operational conditions. At 1500 RPM in loaded condition, both methods show poor behavior. In the dynamic run-up scenarios, on the other hand, the benchmark method is very competitive. It shows a higher recall (less missed detections), but its precision is slightly lower (more false alarms). This results in F1 scores in favor of the benchmark method.

3.2.5 Variant 3: Time/Frequency domain

This variant demonstrates a hybrid approach, in which a domain expert expects a fault to manifest itself in the frequency content, without specifying the exact frequency. It enables fault detection in a large range of working conditions. In this case, the framework is implemented to detect deviations in the spectrogram, a time/frequency domain representation.

[4]This work was carried out prior to the development of the similarity-based anomaly score.

Table 3.6: Performance of the proposed framework for fault detection using electrical signature analysis via harmonic amplitudes. Scores corresponding to a scenario's highest F1-score are indicated in bold. For reference, the last column shows the performance of the classic diagnostics approach (Table 3.3) with the highest F1-score. In five out of the six cases, the proposed approach obtains the best F1 score.

		thr_{cc}						Classic diagnostics approach
	Metric	0.5	0.7	0.8	0.85	0.9	0.95	
820 RPM load	Precision	0.097	0.287	0.865	**1.000**	1.000	1.000	*0.334*
	Recall	0.857	0.857	0.857	**0.857**	0.643	0.045	*0.758*
	F1	0.174	0.430	0.861	**0.923**	0.783	0.085	*0.464*
820 RPM no load	Precision	0.087	0.307	0.895	**1.000**	1.000	1.000	*0.297*
	Recall	0.855	0.855	0.855	**0.855**	0.627	0.009	*0.770*
	F1	0.158	0.452	0.874	**0.922**	0.771	0.018	*0.429*
1500 RPM load	Precision	**0.098**	0.118	0.250	1.000	1.000	1.000	*0.164*
	Recall	**0.855**	0.309	0.009	0.000	0.000	0.000	*0.523*
	F1	**0.176**	0.171	0.018	0.000	0.000	0.000	*0.250*
1500 RPM no load	Precision	0.097	0.250	0.712	0.887	0.979	**1.000**	*0.283*
	Recall	0.855	0.855	0.855	0.855	0.855	**0.855**	*0.796*
	F1	0.175	0.387	0.777	0.870	0.913	**0.922**	*0.418*
Run up load	Precision	0.100	0.535	**1.000**	**1.000**	1.000	1.000	*0.465*
	Recall	0.852	0.852	**0.741**	**0.741**	0.444	0.259	*0.815*
	F1	0.179	0.657	**0.851**	**0.851**	0.615	0.412	*0.592*
Run up no load	Precision	0.100	0.218	**1.000**	1.000	1.000	1.000	*0.427*
	Recall	0.857	0.857	**0.607**	0.429	0.429	0.000	*0.737*
	F1	0.179	0.348	**0.756**	0.600	0.600	0.000	*0.540*

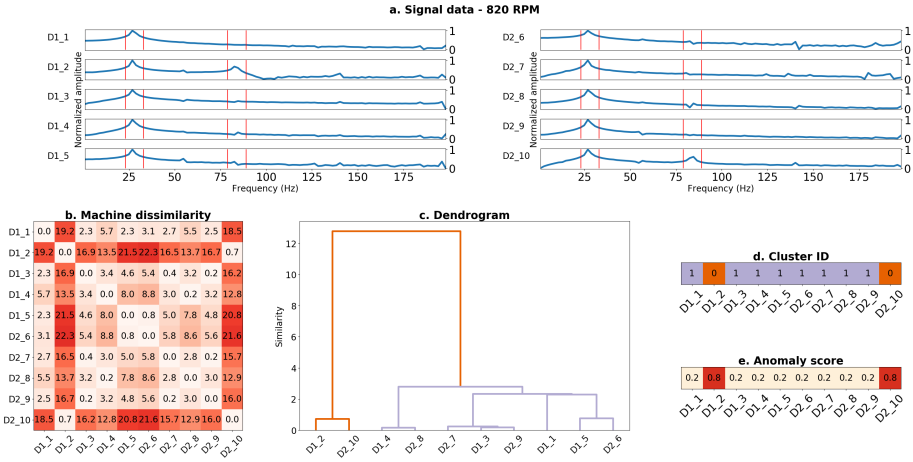


Figure 3.8: Visualization of frequency domain signals (a.), machine similarities (b.), clustering (c. & d.) and anomaly scores (e.). The red vertical line in (a.) indicates the small windows of 10Hz around the fundamental and third harmonic frequencies. Its amplitudes are determined by each window's maximum amplitude. Clusters are partitioned with $thr_{cc} = 0.9$. Each machine is running stationary at 820 RPM. The framework correctly identifies D1_2 and D2_10 as anomalous.

Block 1: Machine comparison

Each machine is represented by its spectrogram, a sequence of FFTs obtained from consecutive windows. This implementation converts the data on a log scale and normalizes using min/max scaling FFT to have similar value ranges for each machine. A low-pass filter at 200 Hz removes unrelated high-frequency content.

DTW (Equation 2.3) is preferred as dissimilarity measure over Euclidean distance, as it is less sensitive to small variations in operational conditions. If the drivetrain's speed fluctuates, DTW will optimally align the spectrograms. In that case, these are considered as n-dimensional time series, with n the number of frequency bins. Each of these bins is equally weighted in this implementation. If desired, a user could opt to have different weighting for frequency bins of interest.

A high-frequency resolution can result in false predictions due to the curse of dimensionality [108]. In that case, noise over unaffected frequencies would be larger than the amplitude increase in a single frequency bin. The task is

however only to perform fault detection. Since the exact frequency at which a deviation occurs is not of interest, it is safe to have a low-frequency resolution. In this variant, the selected FFT window size is 0.05s.

Blocks 2 – 4: Fleet clustering, Anomaly detection & Visualization

This variant reused the default clustering, anomaly detection, and visualization implementations. The first block uses hierarchical clustering and cophenetic correlation. Therefore, the framework is evaluated for various values of thr_{cc} to show the parameter's impact on predictive performance. The second considers the fraction of machines outside each cluster as anomaly scores[5]. A score above $2/3$ is considered as anomalous behavior. Finally, the visualization presents the signal data as a spectrogram.

Results

Figure 3.9 shows the evaluation of a single analysis window. A low-resolution spectrogram shows different machine behavior for faulty machines D1_2 and D2_10. Moreover, the dendrogram suggests using a lower value for thr_{cc}, as the healthy machine D2_7 appears in its own cluster.

The evaluation procedure uses performance metrics to evaluate the framework's ability to detect deviating spectrograms (Table 3.7). It outperforms the benchmark method (Table 3.3) in most stationary scenarios. None of the methods detect an unbalance at 1500 RPM in a loaded condition. In the dynamic run-up scenarios, on the other hand, both methods have similar results. In general, this variant offers the advantage that it does not require domain knowledge or handcrafted rules.

3.3 Application 2: Machine health monitoring through vibration signature analysis

This application involves vibration measurements of an electrical machine fleet, similar to application 1. The task is to detect a voltage unbalance fault based on accelerometer signals. Accelerometers are often already present for condition monitoring of other (mechanical) components, hence it is valuable if they can also be used for voltage unbalance detection. This section illustrates that it is

[5]This work was carried out prior to the development of the similarity-based anomaly score.

Table 3.7: Performance of the proposed framework for fault detection using electrical signature analysis via spectrogram analysis. Scores corresponding to a scenario's highest F1-score are indicated in bold. For reference, the last column shows the performance for the classic diagnostics approach (Table 3.3) with the highest F1-score. In five out of the six cases, the proposed approach has the highest F1 score.

		thr_{cc}						Classic diagnostics approach
	Metric	0.5	0.7	0.8	0.85	0.9	0.95	
820 RPM load	Precision	0.157	0.505	0.738	0.906	**0.990**	1.000	*0.334*
	Recall	0.857	0.857	0.857	0.857	**0.857**	0.768	*0.758*
	F1	0.265	0.636	0.793	0.881	**0.919**	0.869	*0.464*
820 RPM no load	Precision	0.236	0.913	**1.000**	**1.000**	1.000	1.000	*0.297*
	Recall	0.855	0.855	**0.809**	**0.809**	0.618	0.091	*0.770*
	F1	0.370	0.883	**0.894**	**0.894**	0.764	0.167	*0.429*
1500 RPM load	Precision	**0.106**	0.198	0.000	1.000	1.000	1.000	*0.164*
	Recall	**0.627**	0.155	0.000	0.000	0.000	0.000	*0.523*
	F1	**0.182**	0.173	0.000	0.000	0.000	0.000	*0.250*
1500 RPM no load	Precision	0.208	0.839	**1.000**	1.000	1.000	1.000	*0.283*
	Recall	0.855	0.855	**0.855**	0.636	0.118	0.000	*0.796*
	F1	0.335	0.847	**0.922**	0.778	0.211	0.000	*0.418*
Run up load	Precision	0.178	0.487	**0.864**	1.000	1.000	1.000	*0.465*
	Recall	0.852	0.704	**0.704**	0.630	0.593	0.333	*0.815*
	F1	0.295	0.576	**0.776**	0.773	0.744	0.500	*0.592*
Run up no load	Precision	0.221	0.900	0.900	**1.000**	1.000	1.000	*0.427*
	Recall	0.679	0.643	0.643	**0.643**	0.393	0.000	*0.737*
	F1	0.333	0.750	0.750	**0.783**	0.564	0.000	*0.540*

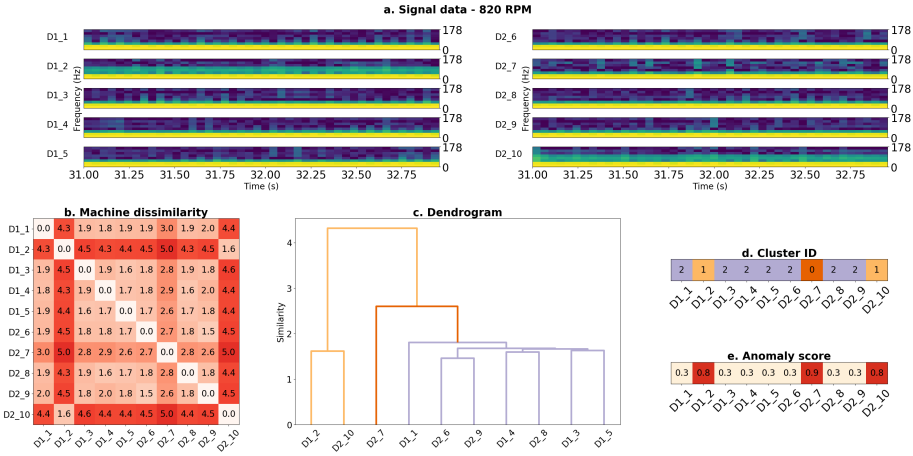
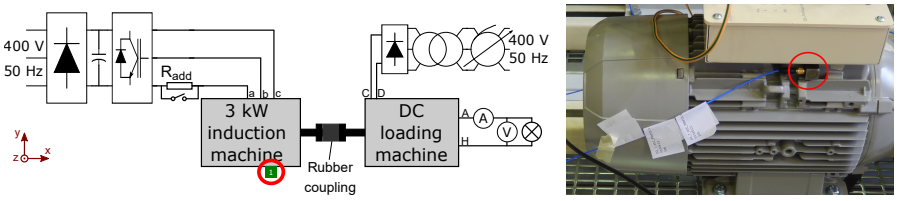


Figure 3.9: Visualization of time/frequency domain signals (a.), machine similarities (b.), clustering (c. & d.) and anomaly scores (e.). The frequency resolution in (a.) is course, to avoid the curse of dimensionality while clustering. Clusters are partitioned with $thr_{cc} = 0.9$. Each machine is running stationary at 820 RPM. The framework correctly identifies D1_2 and D2_10 as anomalous and incorrectly partitions D2_7 by itself. This suggests that the choice of thr_{cc} might not be optimal.

indeed feasible and shows how the framework can use multiple sensor channels and indicators of interest.

3.3.1 Experimental fleet test rig and data set

This application considers a setup of ten electrical drivetrains. A voltage unbalance is again emulated with an external resistor. PCB ICP AC accelerometers measure vibrations. One sensor is mounted on the side of the drive side motor (Figure 3.10) and is sampled at 12800 Hz. Operational speed is measured with a single tachometer mounted on D1_2, triggered once every rotation. This is representative of the full fleet, as all drivetrains are set up to always have the same speed, controlled by an ABB drive controller. One exception is machine D2_8, whose run-up parameters are set incorrectly. For this reason, this drivetrain is not used in the evaluation procedures when considering a run-up.



(a) Illustration of drivetrain D1_2. (b) Drive side motor.

Figure 3.10: A 3D accelerometer (indicated in red) is mounted on the drive side motor and is sampled at 12800Hz.

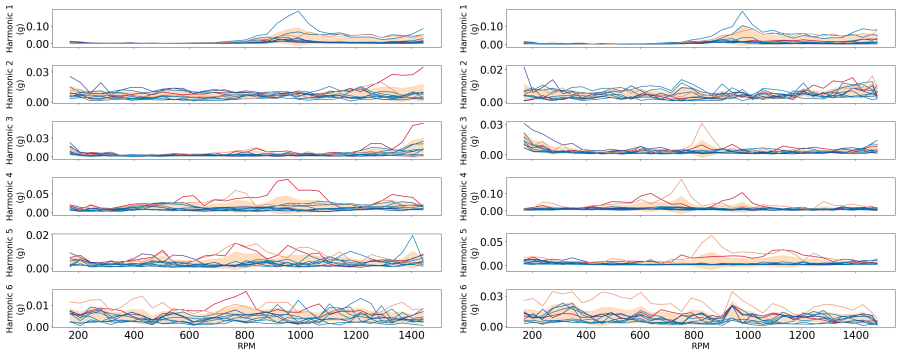
3.3.2 Classic voltage unbalance diagnosis techniques

For a healthy, symmetrical, motor, electrodynamic forces have no alternating components. In the balanced case, the forces are rotating with the flux. In the case of unbalance they have a component rotating in the opposite direction. This results in a pulsation of the amplitude of the forces at twice the stator frequency. This causes radial vibrations in the induction motor [106, 109, 110]. The forces have a tangential component (creating the torque) but also a radial component (the stator flux attracts the rotor like an electromagnet) generating the vibrations.

Detecting voltage unbalance based on vibration signatures is an example of a condition monitoring case that requires different thresholds, as operational conditions affect vibration amplitudes. For example, additional load or operating at a resonance frequency leads to an increase, while lower operating speeds result in lower vibration levels (Figure 3.11). Moreover, many frequencies and sensor directions can be considered. Detailed machine knowledge is required to know these frequencies a priori.

The machine response varies for D1_2 and D2_10, the faulty drive trains (Figure 3.11). Their resonance frequencies differ and in loaded conditions, D2_10 is significantly more affected by the fault compared to D1_2. The optimal indicator depends thus not only on operational conditions but also on the evaluated machine.

The remainder of this use case utilizes a simple condition monitoring method as a benchmark to validate the proposed framework. This method considers a machine as healthy when the amplitude of harmonics 3 till 6, as defined by Formula 3.2, are within σ standard deviations of the fleet's mean. If one harmonic is outside this range, the machine is predicted as being faulty. Note that this is a simplified fleet approach, which avoids setting thresholds manually.



(a) Without load

(b) With load

Figure 3.11: Harmonics 1 (top) – 6 (bottom) of the X-axis accelerometer in a run-up. The faulty drivetrains are shown in blue (D1_2) and red (D2_10). The optimal indicator to detect the voltage unbalance depends on the machine of interest and the operational speed. The orange band indicates the range in which a machine is considered healthy, being 2 standard deviations around the mean.

Table 3.8 shows the fault detection performance for this methodology with different values of σ, Figure 3.11 illustrates the specific case for $\sigma = 2$.

3.3.3 Variant 1: Frequency domain

This variant demonstrates the use of multiple sensors. The implementation combines each of the accelerometer directions (X, Y, Z) in the comparison measure. Moreover, it considers domain knowledge by multiple harmonic frequencies for each of these directions. A data-driven approach automatically selects the best indicator value.

Block 1: Machine comparison

Machines are represented by harmonic frequencies 3 – 6 of each sensor direction. The implementation uses the tachometer to define the fundamental frequency. An FFT converts the time-domain measurements to frequency spectra, in which the maximum value within a window of ±5Hz around the estimated frequencies is considered as their amplitude. Together, these form a sequence with length 1 and 12 dimensions (3 sensor directions, 4 harmonics).

Table 3.8: Performance of classic voltage unbalance detection using accelerometer harmonics 3 – 6. Scores corresponding to a scenario's highest F1-score are indicated in bold.

		σ							
	Metric	0.5	1	1.5	2	2.5	3	3.5	4
820 RPM load	Precision	0.110	0.143	0.168	0.191	**0.201**	0.201	0.201	0.201
	Recall	0.871	0.871	0.817	0.784	**0.772**	0.708	0.653	0.607
	F1	0.195	0.246	0.279	0.307	**0.319**	0.314	0.308	0.302
820 RPM no load	Precision	0.104	0.116	0.125	**0.132**	0.133	0.133	0.133	0.133
	Recall	0.871	0.753	0.648	**0.569**	0.526	0.483	0.445	0.414
	F1	0.185	0.201	0.210	**0.214**	0.212	0.208	0.205	0.201
1500 RPM load	Precision	0.098	0.101	**0.107**	0.107	0.106	0.106	0.106	0.106
	Recall	0.847	0.730	**0.623**	0.510	0.463	0.425	0.392	0.364
	F1	0.176	0.177	**0.183**	0.176	0.173	0.170	0.167	0.165
1500 RPM no load	Precision	0.098	0.107	0.115	**0.119**	0.119	0.119	0.119	0.119
	Recall	0.871	0.837	0.736	**0.620**	0.564	0.517	0.477	0.443
	F1	0.176	0.189	0.199	**0.199**	0.196	0.193	0.190	0.187
Run up load	Precision	0.246	0.301	**0.349**	0.367	0.368	0.368	0.368	0.368
	Recall	0.817	0.756	**0.705**	0.611	0.558	0.512	0.473	0.439
	F1	0.378	0.431	**0.467**	0.458	0.444	0.428	0.414	0.400
Run up no load	Precision	0.249	0.295	**0.322**	0.332	0.332	0.332	0.332	0.332
	Recall	0.848	0.789	**0.670**	0.562	0.511	0.469	0.433	0.402
	F1	0.385	0.429	**0.434**	0.418	0.403	0.389	0.376	0.364

Accelerometer signals can be noisy and require an outlier-robust normalization procedure. This variant uses percentile scaling, as data can be affected by outliers. In this case, normalization also makes sure to equally weigh every harmonic. Otherwise, a harmonic with a larger range could dominate the clustering.

The Euclidean distance measures the differences between a machine pair (Equation 2.1).

Blocks 2 – 4: Fleet clustering, anomaly detection & visualization

This variant reuses the default clustering, anomaly detection, and visualization blocks. The first combines hierarchical clustering with a cophenetic correlation procedure. Therefore, the framework is evaluated for various values of thr_{cc} to show the parameter's impact on predictive performance. The second uses the fraction of machines in each cluster as the anomaly score[6]. A score above $2/3$ is considered as anomalous behavior. Finally, the frequency spectra of each drivetrain are shown combined with the considered harmonic windows.

Results

Figure 3.8 shows the evaluation of a single analysis window. The anomalous behavior of D1_2 and D2_10 is successfully detected in each block.

The evaluation procedure uses performance metrics to evaluate the framework's ability to detect faults based on multiple sensor directions and harmonics (Table 3.9). In loaded scenarios, it outperforms the benchmark method (Table 3.8). Both approaches show poor results in unloaded scenarios, which can be explained by the reduced vibrations in the absence of a load. At 1500 RPM, none of the methods detect the fault due to a decrease of the flux. Lower general performance (compared to electrical signature analysis, Section 3.2) in dynamic run-up scenarios are caused by the fault not manifesting itself in the vibration signature at every operational speed. In general, this implementation offers the advantage that it can consider multiple harmonics, without needing individual threshold values. This is an advantage especially when the optimal indicator varies with the operational conditions.

[6]This work was carried out prior to the development of the similarity-based anomaly score.

Table 3.9: Fault detection performance using vibration signature analysis via harmonic amplitudes. For reference, the last column shows the performance for the classic diagnostics approach (Table 3.8) with the highest F1-score. In five out of the six cases, our framework has the highest F1 score.

		thr_{cc}						Classic diagnostics approach
	Metric	0.5	0.7	0.8	0.85	0.9	0.95	
820 RPM load	Precision	0.121	0.462	0.545	0.565	0.610	**0.908**	*0.201*
	Recall	0.871	0.871	0.871	0.871	0.871	**0.871**	*0.772*
	F1	0.213	0.603	0.671	0.686	0.718	**0.889**	*0.319*
820 RPM no load	Precision	0.097	0.153	0.185	0.196	**0.205**	0.163	*0.132*
	Recall	0.629	0.452	0.435	0.435	**0.435**	0.266	*0.569*
	F1	0.168	0.229	0.260	0.270	**0.278**	0.202	*0.214*
1500 RPM load	Precision	**0.091**	0.004	0.000	0.000	0.000	0.000	*0.107*
	Recall	**0.508**	0.008	0.000	0.000	0.000	0.000	*0.623*
	F1	**0.154**	0.005	0.000	0.000	0.000	0.000	*0.183*
1500 RPM no load	Precision	0.105	0.133	**0.178**	0.172	0.116	0.000	*0.119*
	Recall	0.702	0.435	**0.379**	0.258	0.129	0.000	*0.620*
	F1	0.183	0.204	**0.242**	0.206	0.122	0.000	*0.199*
Run up load	Precision	0.345	0.710	**1.000**	1.000	1.000	1.000	*0.349*
	Recall	0.857	0.786	**0.696**	0.536	0.536	0.429	*0.705*
	F1	0.492	0.746	**0.821**	0.698	0.698	0.600	*0.467*
Run up no load	Precision	0.240	**0.508**	0.469	0.469	0.200	0.143	*0.322*
	Recall	0.857	**0.589**	0.411	0.411	0.107	0.071	*0.670*
	F1	0.375	**0.545**	0.438	0.438	0.140	0.095	*0.434*

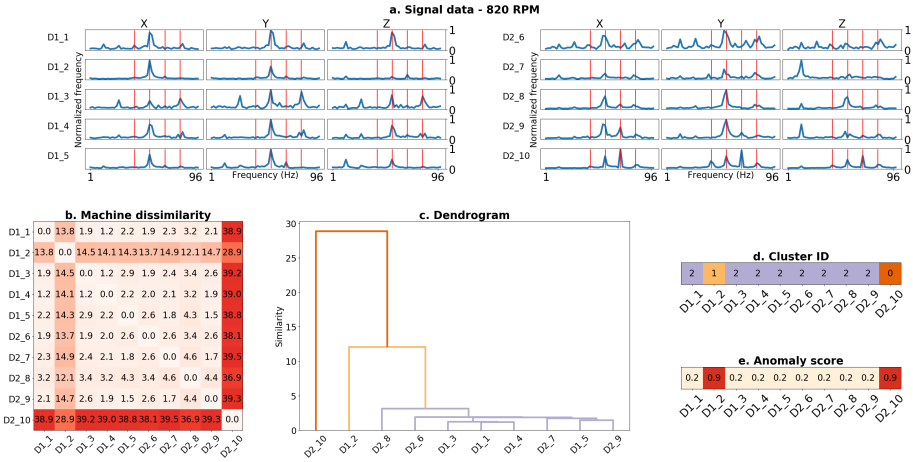


Figure 3.12: Visualization showing the three accelerometer frequency domain signals (a.), machine similarities (b.), clustering (c. & d.) and anomaly scores (e.). The red vertical lines in (a.) indicate the used harmonics 3 – 6. Clusters are partitioned with $thr_{cc} = 0.9$. Each machine is running stationary at 820 RPM, with D1_2 and D2_10 having a voltage unbalance. D1_2 is less affected by the voltage unbalance compared to D2_10, which is visible in the dendrogram.

3.4 Application 3: End-of-production-line testing

In this section, we study the health monitoring framework's applicability for end-of-production-line testing, which involves validating the integrity of newly manufactured products. This is a sequential process, which conflicts with the framework's online comparisons. To overcome this mismatch, we compare the current product being tested to a short-term historical database of recently tested products. The current product is predicted to be faulty if it is considered to be anomalous w.r.t. the recently tested products. After testing, the unit is added to the historical database to support future analysis.

Following the fleet monitoring paradigm, we consider the historical database and the unit being tested as a fleet. This could, for example, involve all tests for a single production batch. Moreover, the end-of-production-line testing application typically satisfies the framework's two main assumptions. First, in most manufacturing settings, production faults are rare. This makes our assumption of a healthy state for the majority of the units realistic to make. However, this also implies that fleet-based analysis might be challenging in scenarios with many systematic failures. We further analyze this in Section 3.4.6,

by studying the minimal number of healthy units required. Second, end-of-production-line tests are generally automated. It is thus easy to guarantee comparable testing conditions for each unit such that signature deviations relate to differences in health status.

3.4.1 Experimental setup

We test the end-of-production-line application by mimicking a gear manufacturer's end-of-production-line test. At the end of the production line, the manufacturer tests the newly manufactured gear by analyzing its transmission error in a dedicated test setup to detect possible errors. In literature, measuring the transmission error is proposed as an alternative to vibration measurements, since it can easily be obtained by using digital encoders [111, 112].

Simulation model We emulate a precision gear test rig, inspired by the physical setup presented in [113] and [114]. Variation within the production line is mimicked by varying the gear pair's testing load. The simulation models the gear's, shaft's, and bearing's deflection by a multi-body approach, which is computationally efficient. This model makes use of three main components. First, we customize the gear pair using the design parameters of Table 3.10 and simulate their meshing with a multi-body gear mesh model. This multi-body model combines the ISO tooth pair's stiffness with a novel slicing technique [115]. This is computationally more efficient compared to a FEM approach [116] and accurate enough to capture the stiffness variation during the mesh cycle, given its use in parametric studies on tooth microgeometry; misalignments; and noise, vibration, and harshness (NVH). Second, we use a finite element model (FEM) approach to simulate the shaft's deflection using a hexahedral mesh, which has better accuracy and reliability compared to an automatically generated tetrahedral mesh [117]. Finally, to account for the bearing's deflection, we used special connectors that define the damping and stiffness characteristics.

Data set The data set includes gear transmission error time series of 26 simulations under stationary speed and torque, sampled at 1000 Hz and having a length of 4.7 seconds. Among these 26 simulations, 7 pairs are misaligned and represent a faulty gear pair. The speed is set to 10 RPM, while the load is randomly sampled in a range of 90 – 110 Nm to mimic the variance of physical tests.

Table 3.10: Gear design parameters.

	Test gear
Number of teeth	57
Normal module	2.60 mm
Normal pressure angle	20 deg
Tip diameter	154.50 mm
Root diameter	141.70 mm
Face width	23 mm
Contact ratio	1.456

3.4.2 Fleet monitoring framework for end-of-production-line testing

Deployment of the fleet-based framework in an end-of-production-line setting can be challenging when no prior data is available. In such a case, the historical database is empty and the framework lacks historical data to make accurate predictions. Due to this low number of comparable units, the prediction accuracy is low until sufficient data is available. In machine learning, this is called the *cold-start* problem, analogous with the mechanical terminology. While a motor needs a certain temperature for optimal performance, the machine learning framework requires data for good prediction performance.

This data can be obtained during operation (a cold start, *heating a device by using it*) or can be provided prior to using the framework (a warm start, *pre-heating the device*). If available, populating the initial database with a few (healthy) units could significantly increase the performance during the start-up phase. These units could for example originate from manually inspected production prototypes. Alternatively, one could first test multiple units and simultaneously analyze them after enough tests are collected. Since this is similar to operational testing, we do not separately discuss this approach. In this work, we analyze the framework's performance in both cold- and warm-start scenarios.

3.4.3 Block 1: Gear comparison

We represent the gear transmission errors time series using the 21 statistical features listed in Table 3.11, and employ Euclidean distance to compare gears. Robust outlier normalization of the features ensures these to be equally weighted

by Euclidean distance. We opt for a normalization inspired by the Z-score: $\frac{X-\tilde{X}}{P_{75\%}(X)-P_{25\%}(X)}$.

3.4.4 Block 2–4: Fleet clustering, Anomaly Detection & Visualization

We use hierarchical clustering combined with partitioning based on cophenetic correlation that constructs a partition using $thr_{cc} = 0.9$. Our similarity-based anomaly score uses the gear comparisons and these clusters to detect deviating gear pairs. Finally, we use similar visualizations as in previous applications.

3.4.5 Experiment 1: Applicability for end-of-production-line testing

We use the gear pair data set in an end-of-production-line setting by sequentially testing the gear pairs (units). We compare each unit under test with the historical database, initially populated by $n_{H_{init}}$ healthy units. After testing, the unit under test is appended to this database to support the next unit's analysis. The framework's prediction performance is evaluated on the units which are not part of the initial historical database, the testing set.

To analyze both cold- and warm-start scenarios, we vary the historical database's initial size $n_{H_{init}}$. Moreover, we study a varying portion of healthy machines by considering different numbers of faulty units n_F. The testing set thus consists of the remaining $n_H = 19 - n_{H_{init}}$ healthy units and n_F faulty units. For statistical reliability, we repeat each experiment 20 times and randomize the units' order in each repetition. The results below show median values.

The results shown in Table 3.12 indicate the correct predictions in each scenario involving only the healthy units ($n_F = 0$). This suggests that the framework rarely triggers false alarms in a purely healthy data set. Additionally, the results show that a warm-start requires only a minimal number of healthy examples. Even by providing only two units to the database, the prediction performance of the first units improves. This indicates that the gear faults are severe and that a successful deployment only requires a small historical database. Indeed, the example in Figure 3.13 shows that the faulty gears significantly differ compared to the healthy gears. Interestingly, the experiments indicate a poor precision for a low number of faulty units ($n_F < 4$), mainly due to the severe impact on the precision of a single false positive: missing the sole faulty machine results in a precision of 0. In conditions with a higher n_F, a single false positive causes less impact on the precision.

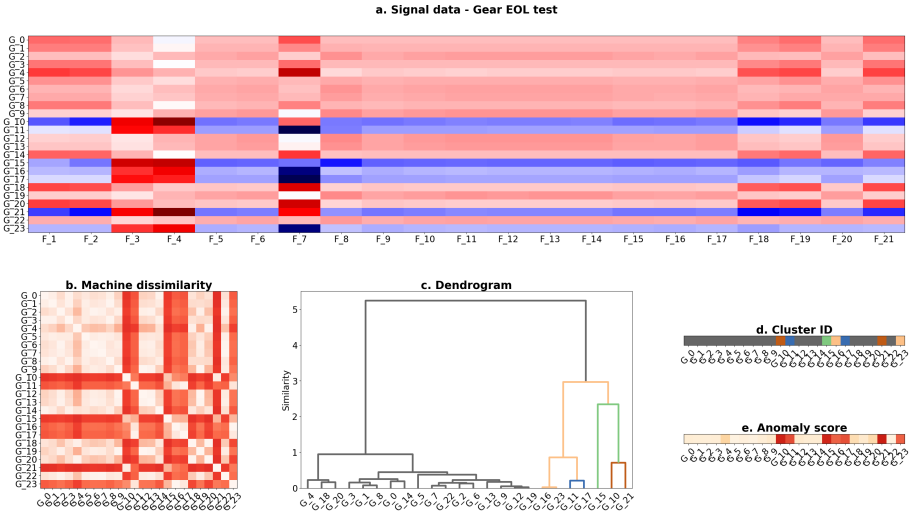


Figure 3.13: Visualization of a gear tests data subset with seven faulty gears (10, 11, 15, 16, 17, 21, and 23). The features are visualized in a blue to red color schema (a.), machine similarities (b.), clustering (c. & d.), and anomaly scores (e.). The features are shown in the same order as in Table 3.11. For formatting purposes, only the feature ID's, colored indications of cluster ID, and anomaly scores on a yellow to red color schema are shown.

3.4.6 Experiment 2: Size of historical data set for warm start

Warm-starting the fleet monitoring framework for end-of-production-line testing requires a historical data set. The required size of this data set depends on the severity of the expected faults.

In this experiment, we study how severe the simulated gear faults are by determining the minimal required number of healthy gear pairs $n_{H_{min}}$ to obtain an F1-score of at least 0.9 in a subset containing n_F faulty units. These units are randomly sampled from the data set. The resulting clusters are partitioned with $thr_{cc} = 0.9$. We consider a unit to be anomalous if $score_{rescale,1.5}(X) \geq 0.5$. For statistical reliability, we repeat this experiment 20 times and report the average values.

The results shown in Table 3.13 indicate that the framework only requires a minimal number of healthy gear pairs with $n_{H_{min}}$ being close to its minimum, $n_F + 1$. Based on the findings of the experiment presented in Section 2.3.3, we conclude that the severity of the gear fault is rather high, causing the healthy

and faulty units to be rather dissimilar. In such a case, increasing the size of the historical data set offers limited performance gains and is thus not required for this application.

3.5 Conclusion

We studied the applicability of our fleet-based condition monitoring framework in two operational health monitoring applications and one end-of-production-line testing application. In both operational health monitoring applications, the framework outperforms the standard condition monitoring approaches. While the framework makes data-driven predictions, we show that it can incorporate and benefit from domain knowledge. Moreover, the end-of-production-line testing application demonstrates that our framework can sequentially analyze machines when a small, short-term historical database is available.

In this work, we only studied applications in which all machines are operating identically. Nevertheless, thoughtful preprocessing, considering external, contextual, or domain-specific information, could allow utilizing the proposed framework in many more applications. For example, a heterogeneous fleet can cause machine diversity. However, expert knowledge such as an ontology could help to select comparable subsets of machines [118, 119, 120]. The framework could then assess each of these subsets individually. Additionally, the machine fleet can operate in dissimilar operational conditions. In such a case, two strategies could allow machine comparisons: normalization and a short-term historical database. The former exploits domain knowledge. For example, if the physics of a machine is available, then it is possible to normalize the measurements with respect to the working conditions such as by removing the effect of speed using angular resampling [31]. The latter contains recent machine measurements in various operational conditions. Clustering operational parameters could provide relevant data samples of machines in comparable operational conditions, which the framework would then use to assess their current health status. Finally, the machine fleet might be subject to different environmental parameters. Clustering machines based on these parameters could result in identifying subsets of comparable machines.

Table 3.11: List of statistical features representing the gear pair data set.

Feature	**Equation**
Amplitude	$max(X) - min(X)$
Crest factor	$\frac{max(\lvert X-\overline{X}\rvert)}{rms(X)}$
Extreme deviation	$max(\frac{max(X)-\overline{X}}{std(X)}, \frac{\overline{X}-min(X)}{std(X)})$
Integration	$\sum_{i=1}^{n} \frac{f(x_{i-1})+f(x_i)}{2}\Delta x_i$ with $\Delta x_i = 1$
Kurtosis	$\overline{(X-\overline{X})^4}/_{std(X)^4}$
Markov regression	$R(1)/_{R(0)}$ with $R(0) = \sum_{i=0}^{n-1} x_i^2$ and $R(1) = \sum_{i=0}^{n-2}(x_i x_{i+1}) + x_0 x_{n-1}$
Mean	$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_1$
Skewness	$m_3/m_2^{\frac{3}{2}}$ with $m_j = \overline{(X - \overline{X})^j}$
Standard deviation	$\sqrt{variance(X)}$
Sum	$\sum_{i=1}^{n} x_i$
Variance	$\overline{\lvert X - \overline{X}\rvert^2}$
Maximum	$max(X)$
Extremum	$max(\lvert X\rvert)$
Median	$\tilde{X}$
Minimum	$min(X)$
Percentile 10	$\%ile_{10\%}(X)$
Percentile 25	$\%ile_{25\%}(X)$
Percentile 75	$\%ile_{75\%}(X)$
Percentile 90	$\%ile_{90\%}(X)$
RMS	$\sqrt{\overline{X^2}}$
Average absolute deviation	$\overline{\lvert X - \overline{X}\rvert}$

Table 3.12: The framework's performance in an end-of-production-line application. Both cold-start and warm-start scenarios are studied by varying the number of healthy units in the initial database ($n_{H_{init}}$). Each row corresponds to a scenario analyzing the n remaining gear pairs among which are n_F faulty.

$n_{H_{init}}$	n_F/n	Precision	Recall	F1-score
0	0/17	1.00	1.00	1.00
	1/18	0.44	1.00	0.58
	2/19	0.79	0.93	0.81
	4/21	0.89	0.93	0.89
	6/23	0.99	0.72	0.80
1	0/16	1.00	1.00	1.00
	1/17	0.42	0.95	0.52
	2/18	0.72	0.93	0.78
	4/20	0.88	0.90	0.87
	6/22	0.96	0.79	0.83
2	0/15	1.00	1.00	1.00
	1/16	0.62	1.00	0.73
	2/17	0.77	0.93	0.81
	4/19	0.90	0.91	0.89
	6/21	0.97	0.91	0.92
4	0/13	1.00	1.00	1.00
	1/14	0.65	1.00	0.74
	2/15	0.82	0.95	0.85
	4/17	0.93	0.94	0.93
	6/19	0.98	0.94	0.96
6	0/11	1.00	1.00	1.00
	1/12	0.79	0.90	0.77
	2/13	0.89	0.97	0.92
	4/15	0.98	0.96	0.97
	6/17	0.99	0.93	0.96
7	0/10	1.00	1.00	1.00
	1/11	0.66	0.90	0.73
	2/12	0.85	0.82	0.81
	4/14	0.95	0.93	0.93
	6/16	0.99	0.93	0.96

Table 3.13: Minimal required number of healthy gear pairs $n_{H_{min}}$ in function of the number of faulty gear pairs. Non-integer values are caused by averaging multiple test for statistical reliability.

n_F	**1**	**2**	**3**	**4**	**5**	**6**	**7**
$n_{H_{min}}$	2.00	3.40	4.50	5.55	6.30	7.10	8.10

Part II: Machine learning with a reject option

The canonical task in machine learning is to learn a predictive model that captures the relationship between a set of input variables and a target variable on the basis of training data. Machine learned models are powerful because after training, they offer the ability to make accurate predictions about previously unseen examples. Since this enables automating a number of tasks that are difficult and/or time-consuming, such models are deployed in a diverse set of application domains.

However, machine learning models' key functionality of always returning a prediction for a given novel input is also a drawback. While the model may produce accurate predictions in general, in certain circumstances this may not be the case. For example, there could be certain regions of the instance space where the model struggles to differentiate among the different classes. Or the current test example could be highly dissimilar to the data used to train the model. In certain application domains, such as medical diagnostics [121] and engineering [122], mispredictions can have serious consequences. Therefore, it would be beneficial for a model to be cautious in situations where it is uncertain about its predictions. In these situations, the prediction task could be deferred to a human expert.

One way to accomplish this is to use machine learning models with a reject option. Such models assess their confidence in each prediction and have the option to abstain from making a prediction when they are likely to make a mistake. This machine learning sub-field was already studied in 1970 by Chow [123] and Hellman [124]. These studies were oriented on the theoretical foundations of the rejection mechanism but laid the fundamentals for future research. Nowadays, the proliferation of machine learning applications has resulted in renewed interest in this area as the ability to abstain from making

a prediction has several benefits. First, by only making predictions when it is confident, the model can offer improved performance on the retained instances [125]. Second, avoiding mispredictions can increase a user's trust in the system. Third, it can still result in time savings by only requiring human interventions to make decisions in a small number of cases. However, the reject option gives rise to new machine learning challenges. These challenges can be categorized in three main areas:

1. Model development
    - Multiple circumstances could lead to mispredictions. Common examples are the lack of training data or the use of an incorrect model family, such as linearly modeling a non-linear relation or deterministically modeling a non-deterministic process. Most machine learning models with a reject option only target a subset of these circumstances. Assessing the applicability of a model with a reject option for the problem at hand thus requires insights in the expected causes of mispredictions in the application at hand.
    - Different architectures to design a model with a reject option exist. For example, the reject option could be completely decoupled from the model's component for making a prediction, or both could be tightly integrated. The former allows reusing existing models, while the latter might lead to better performance.
    - Machine learning consists of discriminative and generative techniques, each having its benefits and drawbacks for both the prediction and rejection components.
2. Model learning
    - Traditional machine learning algorithms are typically not suited to train a model with a reject option. Learning such a model requires a specialized algorithm. This algorithm can either replace the traditional algorithm, or function as an add-on to it. In the former case, the algorithm learns both the prediction and rejection components, while in the latter case it is only designed to learn the rejection component.
    - The risk of abstaining from making a correct prediction needs to be balanced with the risk of making an incorrect prediction. The challenge is to find the optimal trade-off, which is application-dependent. For example, the relative cost of abstaining from making a prediction is different if this would otherwise have led to missing a critical failure in a nuclear power plant or simply recommending a movie to an uninterested user.

3. Model evaluation

    - An ideal model with a reject option only abstains from making an incorrect prediction. However, traditional machine learning performance metrics only study the model's predictive performance do not suffice to measure the model's rejective performance. Instead, specialized metrics are required to adequately evaluate models with a reject option.

The area of machine learning with a reject option is disconnect and lacking unified terminology. Most work is focused on a single application domain [126] or a specific machine learning technique [127]. We found the lack of a clear overview and structure to limit the adoption of machine learning with a reject option in real-world applications. To overcome this gap, we provide a survey aiming to aid both machine learning practitioners and the research community in Chapter 4. In Chapter 5, we demonstrate the applicability of the reject option for operational data analysis by means of a vehicle engineering use-case.

Chapter 4

Survey on machine learning with a reject option

This chapter is based on a publication submitted to Machine Learning Journal:
K. Hendrickx, L. Perini, D. Van der Plas, W. Meert, and J. Davis. Machine Learning with a Reject Option: A survey. 2021

4.1 Introduction

The research on machine learning with a reject option is of significant importance for the adoption of machine learning in several real-world applications. However, in literature, machine learning with a reject option is approached from several perspectives such as statistics [129], information theory [123], and machine learning [130]. This led to disconnect research and terminology. In this chapter, we provide a structured overview on machine learning with a reject option, which we organize around seven key research questions:

Q1. How can we formalize the conditions for which a model should abstain from making a prediction?

Q2. What architectures are possible for operationalizing (i.e., putting this into practice) the ability to abstain from making a prediction?

Q3. How do we learn models that have the ability to abstain from making a prediction?

Q4. Which types of rejections are possible when using standard machine learning models?

Q5. How can we evaluate the performance of a machine learning method with a reject option?

Q6. Where does the need for abstaining machine learning methods arise in real-world applications?

Q7. How does machine learning with a reject option relate to other machine learning research areas?

In addition to the individual contributions of addressing each of these research questions, our major contribution is that we identify the main characteristics of machine learning models with a reject option, allowing to structure the methods in this research field. By providing an overview of the research field as well as deeper insights into the various techniques, we aid to further advance this research area, as well as its adaptation in real-world applications.

The remainder of this chapter is structured as follows. In Section 4.2, we formalize the setting in which machine learning with rejection operates. In Section 4.3, we answer Q1 and identify the two main motivations to abstain from making a prediction. Section 4.4 provides a general overview of how rejectors can be developed and learned, and how rejectors relate to the types of predictors (Q2, Q3, Q4). Section 4.5 introduces means to evaluate the performance of models with a reject option (Q5). In Section 4.6, Section 4.7, and Section 4.8, we provide a more detailed and structured overview on the actionable techniques to reject based on the relevant literature. In these sections, we focus, respectively, on the architecture, the learning phase, and the model type. Section 4.9 treats the main application fields (Q6), while Section 4.10 explores the relation of machine learning with a reject option with other research areas (Q7). Finally, Section 4.11 summarizes our conclusions and lists the main open research questions.

4.2 Preliminaries on machine learning with a reject option

The goal of a machine learning model's reject option is to abstain from making a prediction when a model receives a test example where the risk of making a

misprediction is too large. As these mispredictions typically relate to uncertainty, we further introduce the concept of machine learning with a reject option from a probabilistic perspective. Given a probability space, we indicate with X the multidimensional random variable representing the features, and with Y the target random variable. For the sake of generality, we do not restrict our setting to specific tasks, allowing Y to take values in $\{0, 1\}$ (binary classification), $\{0, \ldots, K\}$ (multiclass classification), $\mathbb{R}$ (regression), among other possibilities. Therefore, we indicate with $\mathcal{Y}$ the general space in which the target variable Y takes values. In the same vein, we indicate the space where the random variable X takes values as $\mathcal{X}$. Let h^* be the unknown function that represents the true relationship between the features X and the target Y, allowing h^* to be either such that $Y = h^*(X)$, when the task is to directly predict the target value from the features, or such that $P(Y|X) = h^*(X)$ when the relation between X and Y is non-deterministic.

The goal of the predictor h is to approximate h^* as accurately as possible, where h is chosen from a certain hypothesis space $\mathcal{H}$ and learned from training data. However, because h only *approximates* h^*, there will likely be regions of the instance space where h significantly differs from h^* and hence h produces inaccurate predictions. The goal of the rejector r is to identify these regions of the instance space. In practice, we want to reject the predictions where the model deviates from h^* by more than a given tolerance error. In classification tasks, this is usually equal to 0, while for regression, a tolerance greater than 0 allows small errors to be made.

Theoretically, a perfect rejector can be seen as a function $r^*\colon \mathcal{X} \to \mathbb{R}$ such that $r^*(x) = \mathcal{L}(h^*(x), h(x))$, where $\mathcal{L}$ is an appropriate given distance function. Because the function h^* is unknown, the problem becomes to approximate the rejector by learning a function $r\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ such that $r(x, h(x)) \leq 0$ (zero is included by convention) when $r^*(x) > \varepsilon$ and $r(x, h(x)) > 0$ when $r^*(x) \leq \varepsilon$, where $\varepsilon \geq 0$ is the tolerance error.

A model with a reject option combines a predictor h and a rejector r. Formally, it is a function $m : \mathcal{X} \to \mathcal{Y} \cup \{\emptyset\}$ such that

$$m(x) = \begin{cases} \emptyset & \text{if the prediction is } \textit{rejected}; \\ h(x) & \text{if the prediction is } \textit{accepted}. \end{cases} \tag{4.1}$$

Consequently, a model with a reject option takes the output of the predictor when the rejector outputs a positive real number, while it takes the symbol $\emptyset$ (meaning rejection) when the rejector determines that the predictor has a large chance of being incorrect. In this last situation, we say that the reject option is active for the given test example. We will derive more formal functions in Section 4.4.1, where the different types of rejectors are analyzed.

Table 4.1: Notation used in this dissertation.

Symbol	Description
X	Random variable representing the features;
x	Vector of features of an example;
$\mathcal{X}$	Space in which X takes values;
Y	Random variable representing the target values;
y	Target value for a given features example x;
$\mathcal{Y}$	Space in which Y takes values;
$\mathcal{H}$	Hypothesis space;
h^*	Underlying true relation between X and Y;
h	Predictor;
r^*	Underlying true reject function;
r	Rejector;
m	Model with reject option;
$\emptyset$	Rejection symbol;
m_r	Rejector component of the model with reject option;
m_h	Predictor component of the model with reject option;
c	Confidence function that outputs a confidence score;
τ	Rejection threshold;
C_c, C_r, C_e	Costs of correct prediction, rejection, misprediction.

4.3 Rejection types

At a high level, a model may be at risk for making an incorrect prediction for two reasons:

1. **Ambiguity rejection.** The learned model h does not faithfully mimic h^* in certain regions of the instance space;
2. **Novelty rejection.** The model h receives a test example that is too dissimilar to any of the examples encountered during training.

Next, we describe these scenarios in more detail and explain why they may arise.

4.3.1 Ambiguity Rejection

Ambiguity rejection allows a model to abstain from making predictions in regions of the instance space where the model cannot recover the correct relationship

between the features and the target variable [131, 123, 124, 132, 133, 134]. One may think that the common machine learning mantra that more training data will enable learning more accurate models will solve this problem. Unfortunately, there are several scenarios where adding more training data will not result in a more accurate model. First, there may be regions of the instance-space where the relationship between the features and the target variable is non-deterministic. Figure 4.1a illustrates this for a binary classification setting where there is a region of the instance space where positive and negative examples are overlapping. Similarly, Figure 4.2a illustrates a regression problem where there is a region of the instance space where the target variable's value is highly variable. One way to interpret this issue is that the chosen feature space does not allow for accurately determining the value of the target value (e.g., some features are missing).

Second, the true target function h^* may lie outside the chosen hypothesis space $\mathcal{H}$. Figure 4.1b illustrates this in a binary classification problem in which a linear model (e.g., a perceptron) is chosen when the true target concept is non-linear. Having access to more data will not solve this problem and in some regions of the instance space, the model will make mispredictions due to this mismatch [135]. Mistakes arising from this mismatch are sometimes referred to as the *bias* component in a bias-variance decomposition of the error. Similarly, Figure 4.2b illustrates a regression problem where a quadratic model is used while the true relation is non-quadratic. While considering a more expressive hypothesis space may decrease such error, finding the most suitable class of models for a task requires domain knowledge. Moreover, more complex models are more computationally expensive to train (e.g., Neural Networks). In addition, they often require more data to learn an accurate model and tend to be more prone to overfitting. These drawbacks make it difficult or undesirable to consider a more expressive model.

4.3.2 Novelty Rejection

Novelty rejection, which is also called outlier [136] or distance rejection [137], allows a model to abstain from making a prediction on an example that is very different from those examples contained in the training set [138, 139, 140]. It is well known that many machine-learned models struggle when forced to extrapolate to regions of the instance space that were not present in the training data [141]. Figures 4.1c and 4.2c illustrate this for a classification and a regression problem. In both cases, the three black stars represent test examples that are far away from the training examples. The learned model h may be inaccurate in these regions due to the lack of training data, which increases the chance of making a misprediction for these test examples.

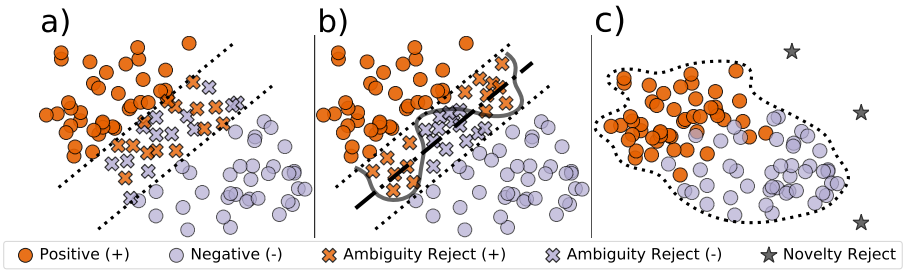


Figure 4.1: Illustration of a classification scenario where the two classes overlap in a region. The dotted lines represent the rejector, the dash-dotted line the fitted predictor h, and the solid line the ground truth relation f. a) shows ambiguity rejection due to a non-deterministic relation between X and Y, b) introduces ambiguity rejection due to the model bias, and c) illustrates an example of novelty rejection. While in the first two plots the rejection region is inside the two dotted lines (examples with cross marks are rejected), in the third figure the rejected novel examples (stars) are outside the dotted line.

There are several reasons why test examples may differ from the training data. First, the training set does not contain examples for each imaginable situation. Second, there may be artificial examples that are deliberately constructed to mislead the classifier (e.g., adversarial examples [142, 143, 144]). Third, the new examples may simply be outliers [145, 146, 147], which can arise due to, e.g., broken sensors. Finally, there may be drift that leads to the emergence of novel data [148, 149].

4.4 Properties of a rejection model

Machine learning models with a reject option can be investigated from three different perspectives: (i) the architecture, i.e., the relation between predictor and rejector at prediction time, (ii) the learning approach, i.e., the methods used to learn the predictor and rejector, and (iii) the model type, i.e., how the choice of type of machine learning model can affect the type of rejection. In the next sections, we explore these categories by providing an answer to the following of our research questions:

Q2. What architectures are possible for operationalizing the ability to abstain from making a prediction? (Section 4.4.1)

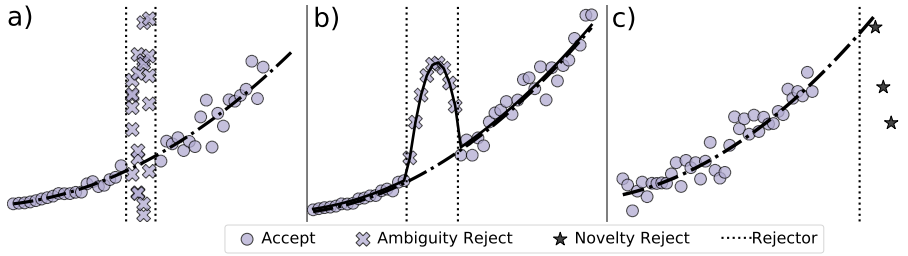


Figure 4.2: Illustration of a regression scenario where rejection can be applied. The dash-dotted line represents the predictor h, while the solid line indicates the true function f. In a), examples in between the dotted lines are rejected (cross mark) due to the high variance of Y. In b), examples indicated with the cross mark are rejected due to the incorrect model bias. In c), star-marked examples are rejected because they differ from the training data (novelties).

Q3. How do we learn models that have the ability to abstain from making a prediction? (Section 4.4.2)

Q4. Which types of rejections are possible when using standard machine learning models? (Section 4.4.3)

4.4.1 Architectures for a model with a reject option

We identify three architectural principles to develop a model with a reject option, based on the data flow between the predictor and rejector at prediction time. From this perspective, the model with a reject option can use a rejector that is a) separated from the predictor, sharing no information at all, b) dependent on the predictor, exploiting information from the predictor such as its output or its properties, or c) integrated with the predictor into a single model.

Separated rejector

In the separated rejector architecture, the rejector operates independently from the predictor. The most typical operationalization of this architecture lets the rejector serve as a filter that decides whether to pass a test example to the predictor. Figure 4.3 shows the data flow of a separated rejector. If the rejector decides to reject the example, then no prediction is made. Otherwise,

the example is passed to the predictor, which returns a prediction. Formally, the model with separated rejector acts as[1]

$$m(x) = \begin{cases} \emptyset & \text{if } r(x) \leq 0 \\ h(x) & \text{otherwise.} \end{cases} \tag{4.2}$$

Using a separated rejector has several benefits. First, the rejector is predictor agnostic. Hence, it can be combined with any type of predictor. Second, because the rejector can be trained independently of the predictor, it is possible to augment an existing predictor with a reject option using this architecture. Third, by serving as a filter, the predictor makes fewer predictions. This is particularly advantageous when there is a high computational cost associated with using the predictor. Fourth, the clear separation between the rejector and the predictor makes it transparent why a prediction was or was not made. Finally, this architecture is generally simpler to operationalize compared to rejectors that interact with the predictor.

However, there are several potential drawbacks. First, not sharing information between predictor and rejector results often in sub-optimal rejection performance because the rejector cannot always take advantage of the predictor's mispredictions [149]. Second, this architecture is typically only used for novelty rejection because, by definition, it can only exploit the properties of an example with respect to the given training data and a novelty rejector only needs to assess how similar the test example is to the training set. Common examples of such novelty rejectors use one-class models or probability density estimators to model the training set.

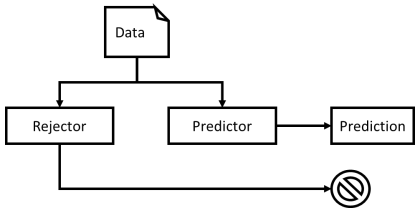


Figure 4.3: Data flow in a separated rejector, in which both the predictor and the rejector are stand-alone models. The rejector is a filter and only passes accepted examples to the predictor.

[1] A rejector r' with a decision value $\tau \neq 0$ can easily be converted to this notation by $r(x) = r'(x) - \tau$

Dependent rejector

In the dependent rejector architecture, the rejector depends on the output of the predictor. Figure 4.4 shows the data flow for the dependent rejector. In contrast to the separated rejector, the dependent rejector considers the predictor's output for the considered test example to decide whether to reject it or not.

Formally, the model with dependent reject option acts as

$$m(x) = \begin{cases} \emptyset & \text{if } r(x, h) \leq 0 \\ h(x) & \text{otherwise.} \end{cases} \tag{4.3}$$

Designing a rejector as an extension on the predictor has several benefits. First, the interaction between the predictor and rejector enables ambiguity rejections. Second, a dependent rejector can extend an existing model with a reject option. Often, this is done based by estimating the prediction uncertainty with a confidence metric derived from the predictor. However, the quality of a dependent rejector is highly influenced by the quality of such metric.

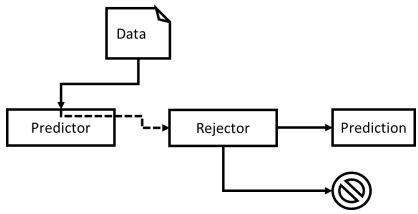


Figure 4.4: Data flow in a dependent rejector. First, the predictor processes the example at hand. Next, the rejector assesses the confidence in the prediction based on the predictor's representation of the example.

Integrated rejector

This architecture integrates the rejector and predictor into a single model by treating the rejection as a special case with an additional output $\emptyset$ (both in classification and regression tasks). Figure 4.5 illustrates this scenario, where the effect of each component cannot be distinguished. Formally, the model with integrated reject option acts as

$$m(x) = \begin{cases} \emptyset & \text{if } m_r(x) \leq 0 \\ m_h(x) & \text{otherwise} \end{cases} \tag{4.4}$$

where m_r and m_h indicate, respectively, the rejector and predictor components of the model m. Note that the effects of the two components are indistinguishable inside the model, and that they affect each other during the learning phase.

This architecture has two key benefits. First, integrating the predictor and rejector means that both aspects of the model are optimized towards the task at hand. This can improve the performance of the model with a reject option when compared to using a separated or dependent rejector architecture, because the predictor's and the rejector's components can affect each other. Second, because it is a unique model, the bias introduced by the model with a reject option is potentially less than the scenario where the predictor and rejector are two different models.

However, designing such a rejector might not be trivial. First, it requires extensive knowledge about the predictor in order to integrate the reject option. Second, it may require developing a novel algorithm to learn the model with a reject option from data.

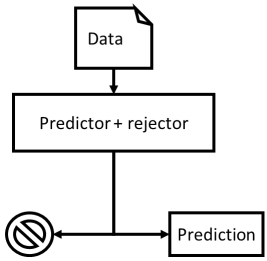


Figure 4.5: Data flow in an integrated rejector, in which the predictor and rejector are one model. This model embeds the reject and predict functions and directly outputs a prediction or a rejection.

4.4.2 Learning perspective

As is standard in machine learning, a model with a reject option needs to be learned from training data. This entails learning both the predictor and the rejector, which overlap when using an integrated architecture. Allowing the model with a reject option to abstain from making predictions when it is likely to make mistakes introduces a clear trade-off: one wants only to make predictions for those examples where high predictive performance is expected, while at the same time making predictions for as many examples as possible. Both extremes – almost always making a prediction but achieving low accuracy and rarely making a prediction but achieving high accuracy – are inadequate

for practical use. Therefore, the learning approach for training a model with a reject option needs to navigate this trade-off between the model coverage (i.e., the proportion of predictions made) and the model performance (i.e., the performance of the model on non-rejected examples) [150, 151].

We distinguish between two techniques for training the predictor and the rejector: sequentially and simultaneously. In the first case, the rejector is usually trained after the predictor, while in the second case they are jointly learned. In the following section, we describe these two learning principles and relate them to the architectures they are relevant to.

Sequential learning

Sequential learning entails using separate approaches to learn the predictor and rejector. This approach is only applicable to the separated and dependent rejector architectures: by definition, it cannot be used with integrated rejectors because this architecture does not distinguish between the predictor and rejector components. When using a dependent rejector, the predictor is typically learned first because learning the rejector often entails ascertaining when the predictor is likely to be inaccurate. The sequence of learning is not important for the separated rejector architecture.

This learning approach is popular because it allows extending many existing predictive models to include a reject option, or even to reuse a previously-learned model. For example, a reject option can be added to a Support Vector Machine (SVM) by rejecting examples that are farther from the decision boundary than a specified value because these may represent novel examples that are dissimilar to those in the training set [146].

A drawback to the sequential approach is that it could lead to a sub-optimal predictor-rejector combination because there is a limited dependence. That is, the rejector can depend on the learned predictor or vice versa but there cannot be a mutual interaction between their learning approaches. To illustrate this, consider a dependent rejector that rejects examples by setting thresholds on a probabilistic predictor's outputs. Sequential learning requires learning the predictor first on the full dataset and second setting the threshold on its output for rejection. Hence, the predictor cannot benefit by knowing which examples will be rejected. If it knew this, it could exclude those examples from the training data, and thereby improving the quality of its learned model.

Simultaneous learning

Simultaneous learning involves training the predictor and the rejector within a single algorithm. This offers the key benefit of allowing the learning procedure to capture the interplay between the predictor and the rejector. Capturing this interdependence can lead to more accurate predictors and rejectors. For example, if the learner knows which training examples will be rejected, then these can be excluded from the training set. Excluding these examples can improve the predictor's performance by avoiding, e.g., overfitting on outliers, which algorithms such as regressors and decision trees can be sensitive to.

However, an algorithm for simultaneous learning needs to be specifically designed for a given predictor and rejector, as well as for the specific architecture involved, and cannot be simply reused from a different setting.

Moreover, simultaneous learning tends to be computationally more expensive than sequential learning. From a probabilistic perspective and when assuming the predictor and rejector being independent, sequential learning can be achieved by learning $P(Y|X)$ for the predictor and by learning $P(X)$ for the rejector. Simultaneous learning on the other hand requires the more complex task of learning $P(X, Y)$ because $P(Y|X)$ and $P(X)$ cannot be optimized separately and more parameters are required.

4.4.3 Relation between model types and rejection

When employing a standard machine learning model as either a predictor or a rejector, a natural question is whether the model allows for ambiguity rejection, novelty rejection or both. For example, using a probability estimation tree as predictor can only indicate ambiguity rejection since it only contains information on the posterior probability $P(Y|X)$. In contrast, using the Local Outlier Factor (LOF) [152] as a rejector only captures the training set's density $P(X)$, enabling novelty rejections but no ambiguity rejections. Whether we refer to a model as predictor or as rejector depends on the architecture. For a separated rejector, the allowed rejection types are purely determined by the rejector due to the independence between predictor and rejector at test time. For a dependent or integrated rejector, the predictor is responsible for the type of rejection and, therefore, we refer to the predictor instead of the rejector.

Generally, the information a model captures follows from the type of machine learning technique the model is based on. We distinguish two main categories: i) *generative models* which model the observed training set and ii) *discriminative*

models which only focus on the relation $X \rightarrow Y$. Table 4.2 summarizes the main conclusions of this subsection.

Table 4.2: Overview of tasks that can be performed by the different model types.

	Ambiguity rejection	Novelty rejection
Prob. density Generative	x	x
Data density Generative		x
Discriminative	x	

Generative models

The underlying objective of the generative approach is to model the observed training set. Broadly speaking, there are two main ways to accomplish this. First, it is possible to take a *probabilistic approach* and model the joint distribution (density) of (X, Y) over the input space (i.e., the features and target variable). Second, it is possible to model the *data density* of X, which is typically done using distances or similarities among the examples.

Probability-based methods Probabilistic generative models typically model the joint distribution of the input space (i.e., $P(X, Y)$) using some parametric form. The goal is to learn the parameters of the assumed distribution from the observed training data. A canonical example is a Gaussian Mixture Model (GMM), which assumes that the data are being generated by multiple Gaussians. The goal is to learn the optimal values for all Gaussians' moments as well as its mixing weight. However, more complex models like Bayesian networks, Hidden Markov Models (HMMs), or Variational Autoencoderss (VAEs) are widely used as well.

Probabilistic approaches are extremely flexible because it is possible to derive $P(Y|X)$ and $P(X)$ from $P(X, Y)$. This flexibility has two benefits. First, they can serve as either the predictor or rejector in a separated or dependent architecture. Moreover, they can be used as the model in an integrated architecture as well. Second, they perform both novelty and ambiguity rejections. Novel examples correspond to those that have a low probability of being generated, $P(X)$. Ambiguity rejection is possible by, e.g., setting thresholds on the inferred $P(Y|X)$.

While the flexibility of probabilistic generative approaches makes them seem ideally suited for learning with rejection, unfortunately estimating $P(X, Y)$ is

often hard because of two reasons. First, it requires modeling the relations among all features X. This is especially challenging in high dimensional problems, as it would necessitate extremely large training sets or making strong assumptions (e.g., the conditional independence of Naive Bayes) [153, 154]. Second, $P(X, Y)$ models the entire instance space instead of focusing on the most relevant areas for the task. For example, for prediction and ambiguity rejection, only the regions near the decision surface are of interest whereas novelty rejection only requires identifying sparse regions of the instance space [155]. Being too general may degrade the model's performance [156].

Modeling data density Modeling the data density entails determining which regions of the instance space are densely (i.e., contain many training examples) and sparsely (i.e., contain few training examples) populated. Typically, these approaches take a local approach to estimate $P(X)$ and for a specific example look at how many other training examples are in its neighborhood. That is, how many other examples are nearby or similar to the example of interest. This approach is frequently employed by anomaly detection algorithms such as the k-nearest neighbor outlier algorithm [157] or the local outlier factor [152]. Clustering algorithms such as DBSCAN [158] also try to exploit data density. One-class classifiers, like One-Class Support Vector Machine (OCSVM) [159] can also be interpreted through the prism of modeling the data-density as they determine a boundary that encapsulates the training data. In this survey, data density is considered to be a generative model because it allows sampling novel examples that are similar to the original training data (e.g., using Markov chain Monte Carlo algorithms or rejection sampling [160]).

Density-based approaches' primarily use is to serve as the rejector when performing novelty rejections: if a test example falls in a sparsely populated region with respect to the training data, then it is rejected. Unlike the probabilistic approach, they do not make strong assumptions about the form of the density. However, unlike probabilistic models, they typically do not allow recovering the relationship between the target variable and the features. This makes it difficult to use them for ambiguity rejections.

Discriminative models

Discriminative models focus on capturing the relationship between the features and the target variable. Often, these models focus on determining the decision boundary between the different classes. Classic examples of discriminative models are logistic regression, decision tree classifiers, SVMs and standard multilayer perceptron Neural Networks.

The primary role of discriminative models in a learning with a reject architecture is to serve as a predictor. In this role, they most commonly facilitate performing ambiguity rejection as they often produce a confidence associated with their predictions. For example, logistic regression estimates $P(Y|X)$ or an SVM returns an example's distance to the separating hyperplane. On the other hand, most discriminative models do not contain information about $P(X)$. Hence, it is not possible to use a discriminative model to determine whether an example deviates from the training set and hence, to perform novelty rejection.

4.5 Evaluating models with a reject option

A model with a reject option has two goals. On the one hand, it wants to have high accuracy, that is, it is as accurate as possible when it does make a prediction. On the other hand, it wants to have high coverage, that is, it should make a prediction for as many test examples as possible [161, 151]. Unfortunately, these two goals are competing as the accuracy can be increased by simply only making a prediction in the most confident cases, which reduces coverage [150, 162]. Metrics specifically tailored to the learning with a reject option are needed to capture this trade-off.

Similar to standard machine learning, evaluating the reject option's quality requires the test set to include examples relevant to the rejection type of interest. This might be problematic for novelty rejection as novel examples often arise once the model is deployed. Instead, such rejectors can be validated based on artificially generated data points or a withheld subset of the training data, such as a specific class.

Broadly speaking, two categories of metrics exist. The first category encompasses metrics that evaluate models given a fixed rejection rate. In this case, standard evaluations can be used such as the accuracy, F1-score or error rate of the predicted examples. The second category of metrics assesses the performance in a variety of operating conditions to gain insight into its performance-coverage trade-off much like Receiver Operating Characteristic (ROC) or Precision-Recall (PR) analysis would do in standard machine learning. Regardless of the metric, standard machine learning methodology is applicable and performance is evaluated on an independent test data set that is assumed to be representative for the environment the model will be deployed in. This implies that examples are present in the test data that are representative of the rejection type(s) of interest.

4.5.1 Evaluating models with fixed rejection rate

Condessa et al. [163] argue that a good evaluation metric should meet three main criteria: i) it should depend on the model's rejection rate, ii) it should reach its maximum value for a perfect rejector (i.e., a rejector which rejects all misclassified examples) and its minimum value for a rejector which rejects all accurate predictions, and iii) it should allow comparisons of models with different rejection rates.

Additionally, Condessa et al. [163] introduce three types of evaluations and show that these meet the required conditions:

1. **Prediction quality.** In machine learning with rejection, the model's *prediction quality* can be assessed using standard performance metrics on the non-rejected examples such as the accuracy [163, 164, 165] or the F1-score [166, 167, 168]. It is possible to compare different models based on their prediction quality, even if the models have a different rejection rate. However, this is not in line with making a trade-off between the prediction quality and rejection rate.

2. **Rejection quality.** The *rejection quality* evaluates the reject option as a binary classification problem, separating mispredicted examples and novel classes from correctly predicted examples. This analysis is done with typical machine learning metrics such as Area Under the ROC Curve (AUC) [169, 170], F1-scores [169], accuracy, False Negative Rate (FNR)/True Postive Rate (TPR) [170], and precision/recall [163, 170].

3. **Combined quality.** The *combined approach* evaluates the model with a reject option as a whole, measuring the predictor's performance on the non-rejected examples and the rejector's performance on the rejected examples. Therefore, the probability of an example being accepted and correctly classified or rejected and misclassified is used as a quality metric [163, 171]. Alternatively, this quality can be quantified with a cost function accounting for the correct prediction cost C_e (usually 0), misprediction cost C_e, and rejection cost C_r.

4.5.2 Evaluating the model's accuracy/coverage trade-off

A common principle to study a model's accuracy/coverage trade-off is to evaluate the non-rejected prediction quality in function of the model's rejection rate, which is typically varied between $[0\%, 100\%)$. This principle can be applied to several standard performance metrics. An Accuracy-Reject Curve (ARC) [172]

shows the accuracy of the non-rejected examples in function of the rejection rate. Ideally, this curve is as high as possible, indicating a maximal accuracy for each rejection rate. ARCs are widely used for classification tasks [143, 144, 173, 174, 168, 175, 176, 177, 178, 166, 179, 180, 181, 164, 165, 182, 183, 184]. Also alternative measures for the prediction quality can be used on the y-axis instead of the accuracy such as F-scores [167, 185, 186, 187], or the accuracy variation [188, 189, 190]. In regression problems, models are often evaluated using curves similar to ARCs in which the accuracy is replaced by a risk function such as the mean squared error [135, 141, 191, 192, 193]. However, since these curves use a risk instead of a quality measure, these curves are ideally low. Finally, ROC analysis can be extended with a third dimension for the rejection rate [185]. Moreover, similar to AUC in standard machine learning, the Area Under the ARC Curve (AURC) is considered to be indicative for the model overall performance [172, 185].

However, generating these curves can be challenging for two reasons. First, some rejectors do not allow directly setting the rejection rate [149, 178, 194]. Instead, they might have other hyperparameters such as rejection costs [195, 196, 197]. In these cases, one can study the non-rejected prediction performance in function of this hyperparameter [198, 199, 200]. Second, altering the rejection rate of simultaneously learned rejectors can require completely retraining the model with a reject option. This may be too computationally demanding to perform a fine-grained analysis [163].

4.6 Instantiating a model with a reject architecture

When designing a model with a reject option, there are three types of architectures that can be used: separated, dependent and integrated (see Section 4.4.1). In this section, we discuss how these architectures are instantiated throughout the literature and what the relevant design choices are. In addition to the three architectures, in Section 4.6.4 we discuss how researchers have also combined multiple architectures and rejectors to be able to support different types of rejection.

4.6.1 Separated rejector

In the separated rejector architecture, the rejector is a stand-alone model that, at test time, decides whether to reject an example without considering the predictor. A separated rejector can be used for any type of rejection.

For novelty rejection, the rejector in a separated architecture typically determines if a test example is novel by learning a model of the training data. One approach is to employ a one-class classifier and treat finding outliers as a novelty detection task. Common techniques are Support Vector Data Descriptor (SVDD) [194] and OCSVM [149, 201], which model the training data by finding a hypersphere that encapsulates the training data. If a test example falls within the learned hypersphere a prediction is made, otherwise the test example is rejected. Optionally, the hypersphere can be enlarged to increase the predictor's coverage rate [194]. A second approach is to problematically model the training set, for example with a density-based model such as a GMM [148, 202]. A test example is considered to be novel if it has a too low probability of being generating by the distribution that the learned GMM represents. A third approach is to use outlier detectors [201, 203]. In novelty rejection, it is important that the outlier detector is able to extrapolate outside of the training data. Practically, this means it must be able to return an outlier score that is higher (where high scores are associated with outliers) than the highest score of the training examples. Otherwise, the rejection threshold will either reject no instances (if set too high) or reject training data (if set too low). This problem is partially mitigated if the training data already includes outliers as these outliers will receive the highest outlier scores. However, this is difficult to guarantee for an arbitrary dataset and generating outliers artificially has the risk of introducing an unintended bias.

For ambiguity rejection, only a handful of works are available that consider a separated rejector. This is probably because it requires the rejector to replicate the behavior of the predictor without having access to it. Therefore, the other two architectures are better suited for performing this type of rejection. An example where this architecture has been applied for novelty rejections is by Asif and Minhas [193]. They exploit two separate Neural Networks that jointly train a predictor and a rejector using a meta-loss function that penalizes prediction errors as well as incorrect rejections. While trained simultaneously, the rejector is a standalone model that operates independently of the predictor at test time. Hence, the rejector can independently identify test examples that are similar to training examples for which the predictor made a misprediction.

4.6.2 Dependent rejector

At test time, a dependent rejector takes the example-wise predictor's confidence as input to reject predictions with low confidence values. Essentially, there are two degrees of freedom when developing a dependent rejector: the choice of the confidence metric c, and how to define "low confidence values". Usually, the second aspect is tackled by setting or learning a threshold on the confidence

scores such that all the values below it are rejected, while the others are accepted. The main advantage of threshold-based methods is that the confidence scores only need to be consistent and on the correct side of the threshold [204, 205]. Moreover, it is often sufficient that the confidence metric merely produces an accurate ranking of the examples [127].

Confidence metric

A confidence metric represents a predictor's uncertainty in a prediction. What constitutes an appropriate metric depends on the rejection type. For an ambiguity rejection, the metric should be indicative of the variability of the target variable or the potential predictor bias. For a novelty rejection, the confidence should capture the example's similarity to the training data. While some theoretical works assume the confidence scores are given so they can analyze the rejector's idealized behavior [123, 206, 182], in practice these scores need to be derived from the predictor. We distinguish among three methods for deriving confidence scores based on whether they use the predictor's i) hard predictions, ii) soft prediction, or iii) model properties. Although a method that exploits hard predictions is applicable to many different types of predictors, it is likely to be sub-optimal since it does not exploit information specific to the considered predictor. On the other hand, using the predictor's properties is likely more effective, but requires the method to be specific to the chosen predictor. Using soft predictions falls in-between the other two scenarios, which motivates why most work takes this approach.

Confidence metrics based on hard predictions There are two strategies to assess a predictor's confidence based on hard predictions. Both make use of multiple predictions and consider variance in these predictions as an indication for ambiguity rejection.

The first approach uses an ensemble of models [207, 190]. If the models in the ensemble strongly disagree about a test example's prediction, then it is assumed that the examples lie in an ambiguous region and should be rejected [134, 207, 150, 208, 209]. This disagreement can be measured in several ways such as considering the entropy of the predictions. Moreover, the predictions can be weighted uniformly [210] or non-uniformly [211, 212, 213]. Such ensembles have been constructed using: a diverse set of classifiers [214, 215, 216], Neural Networks [217, 218, 219, 135], tree-based predictors [220], prototype-based predictors [221, 222, 218, 223, 124, 136], probabilistic models [224, 218], and k-Nearest Neighbors (kNN) models [192]

The second approach uses a single predictor. It generates multiple different predictions by slightly perturbing the test example. For image data, such perturbations can be geometric functions (like rotations or symmetries) [135, 192]. Depending on the model, this strategy could be applicable to both ambiguity and novelty rejection. First, if the test example lies in an ambiguous region of the instance space, there will be a large variance in the set of predictions [225]. Second, novel examples can be detected by evaluating the entropy of the predictions. Alternatively, the reconstruction performed by an autoencoder can be used to generate variations. Joana et al. [226] feed the reconstructed example to the predictor and contrast the prediction with the prediction for the original example. This works as only instances similar to the training data are assumed to be closely reconstructed.

Confidence metrics based on soft predictions Ad-hoc confidence metrics can be derived from the output of a predictor making soft predictions. We distinguish between i) binary or multiclass classification, ii) multi-label or class-selective classification, and iii) regression tasks.

In binary or multiclass classification, some predictors produce a value representing how likely it is that the example belongs to a class. If these scores are monotonic mappings of the predictor's example-wise confidence, the rejector can use them to decide to abstain from making a prediction for ambiguous and novel examples. Generally, ambiguity rejection is done when, after ranking, the predictor is similarly confident in the top two (or more) classes. Additionally, a relatively low confidence in the top class indicates a novel example.

A widely used approach to obtain confidence scores is to derive posterior probabilities from the predictor's soft predictions by using techniques like calibration functions [179], discriminant functions [227], normalizing the predictor's outputs for the classes [228, 121], or measuring the impurity in the leaves in tree-based methods [229, 189, 180]. On the other hand, certain predictors directly produce confidence scores, such as the posterior probability of Logistic Regression models [230, 231, 232, 233, 234] and Gaussian Mixture Models [235] or the softmax layer of Neural Networks [195, 161, 129, 143, 217, 219]. Additionally, the confidence scores can also be used indirectly and serve as an input to another model. For example, a Neural Network predictor's soft outputs are fed into a kNN classifier to emulate the regions where the Neural Network faces ambiguity [236, 237].

In multilabel classification and class-selective classification, an example can belong to several classes simultaneously. Thus a predictor must return a set of labels. In this setting, a rejection can take two forms. First, it can perform

a partial prediction and reject a subset of the predicted classes. Second, it can perform a total rejection and return $\emptyset$. Therefore, the predictor usually associates one confidence score with each possible class. Because multiple classes can be predicted, these scores should be treated independently. Using multiple confidence scores enables performing a partial rejection by having one rejector associated with each possible class [206, 238, 239, 240]. This becomes a total rejection if no candidate classes remain. Practically, a few approaches to derive confidence scores have been proposed. When Neural Networks are used, the neuron scores of the last layer can be considered as the class-wise confidence scores [241]. Similarly, one can use the purity of the nearest neighbors [186, 168] and tree-based methods [189], as well as probabilities from multiple Logistic Regression models [187]. Additionally, heuristic density-based and distance-based metrics such as normalized symmetrical kernel functions can also be used [242].

Soft predictions can also indicate a regressor's confidence. For example, posterior probabilities estimated from a Neural Network's softmax layer allow ambiguity rejection [191]. Differently, confidences can be derived from the variance of a transductive model, as high variance is an indication for a novel example. For example, Sambu Seo et al. [141] demonstrates this approach based on Gaussian Processes, which quantifies the prediction uncertainty.

Confidence metrics based on predictor's properties Several approaches exist that produce confidences tailored to the specific properties of the predictor. Depending on the used properties, such a confidence value is applicable to ambiguity and novelty rejection. Two primary ways exist for doing this.

The first option is to derive a metric based on the meaning of the predictor's scores. For instance, an SVM's score indicates the distance between the example and the decision boundary. Therefore, a usually high distance is indicative of a novelty whereas a very small distance of an ambiguity [243, 244, 245, 246, 247, 248, 205, 122, 171, 249, 250, 251]. Another approach is to identify novel examples based on on the distance to the k-th nearest neighbor [145, 138] or to the closest prototype [136, 224, 222, 252, 176, 253]. To perform an ambiguity rejection with kNN one can consider the purity of class distribution within a test example's neighborhood set [137, 254, 186, 127, 177, 181, 180]. For prototype-based methods, the relative distances to the two closest prototypes can be considered [136, 224, 222, 252, 176, 255]. Additionally, an example's proximity to an angle-based classifier's decision surface can be measured by means of geometric functions [256].

The second option is to do ambiguity rejection by quantifying the predictor's uncertainty based on its properties. For instance, the posterior probabilities can

be derived from the weight vectors of a Self-Organizing Map (SOM) [257, 209]. Another example is to measure the predictor's disbelief index, measuring how much a predictor deteriorates if it were constrained to predict the example at interest differently [258]. A low disbelief then indicates the model being indifferent to the example's label, which is assumed as an indication for ambiguity. However, this approach is only feasible if (re)training the model is computationally cheap.

Threshold

A standard way to decide whether to accept or reject a prediction is to set a threshold τ on the examples' confidence score: those with a score lower than or equal to τ are rejected and all the others are accepted. Formally, the threshold-based dependent rejector is

$$r(x, h) = c(x, h) - \tau, \tag{4.5}$$

where $c(x, h)$ is the confidence metric that depends on the instance x and the predictor. Based on the definition used in Equation 4.3, an example is rejected if the rejector outputs a negative value. The threshold τ is a hyperparameter that needs to be tuned as its value trades off the predictor's coverage rate and the performance on accepted examples [131, 123, 132, 150, 151, 259].

Typically, a *global threshold* is used, which entails setting a single value for τ. The underlying assumption for using a global threshold is that the predictor's confidence is equally calibrated and accurate over the entire instance space [210, 260, 240, 196, 261, 262, 263, 185, 264, 265, 266]. While being a strict assumption, using a global threshold only requires setting one value. This makes it both simple and transparent.

However, in several cases considering multiple *local thresholds* $\tau_1, \tau_2, \ldots, \tau_v$ (with $v > 1$ indicating the number of thresholds) may be beneficial [223, 267]. Exploiting local thresholds is particularly useful when the confidence metric's accuracy varies across different regions of the instance space, which could be the case for several different reasons. For instance, the variance can be non-uniform over the feature space [177], such as different class-wise variance. Another possibility is that the predictor is more accurate in certain cases [268, 269, 168], such as being more accurate on some classes than others. Finally, mistakes on some classes can have a higher cost [267]. While giving more fine-grained control over a rejector's performance [136, 270, 235], setting multiple thresholds requires more computational effort and data [136]. Fischer et al. [271] cast the problem of finding the best local thresholds as an optimization problem that

minimizes the empirical risk error. They propose an exact approach that runs in polynomial time and an approximate one that is linear.

Instead, tackling variations in the confidence metric by using multiple local thresholds, one can also calibrate the confidence metric to behave consistently such that a global threshold suffices [153, 272]. In addition to limiting the number of thresholds, this approach allows considering a rejector based on multiple confidence metrics where each one captures different aspects of the prediction uncertainty.

4.6.3 Integrated rejector

In an integrated architecture, it is impossible to distinguish between the role of the predictor and rejector at test time. We distinguish between two ways to design an integrated rejector architecture. First, the rejector and predictor are designed together to jointly maximize the performance on the non-rejected examples. Second, the rejector is integrated into the predictor such that rejection becomes part of the prediction process.

Rejector and predictor are designed together The integrated rejector is developed jointly with the predictor's component such that the model with a reject option is fully oriented to the given application task. Typically, this involves a loss function in the learning procedure that considers both prediction accuracy and rejection. In literature, we observe two categories adhering to these criteria, one based on SVMs and one on Neural Networks. The former is focused on ambiguity rejection, while the latter enables both rejection types.

The first category entails methods based on an SVM. However, contrary to SVM-based dependent rejectors, these integrated rejectors do not rely on the distance to the decision surface. Instead, they directly model the ambiguity rejection area by using two hyperplanes instead of one, rejecting examples located in between the two hyperplanes. To learn these two hyperplanes, two strategies have been proposed. One approach alters the SVM learning algorithm to directly learn a pair of hyperplanes, parallel and equidistant to the optimal separating hyperplane [273, 174]. The other approach is to derive the hyperplanes from multiple SVMs, each learned to differently prioritize the classes [245, 244, 197, 167]. We do not consider this to be identical to ensembles used in dependent rejectors, as it makes use of a set of identical models and is specifically designed for rejection. A potential problem with this approach that independently learns the hyperplanes is that it might lead to intersecting hyperplanes, for which rejection is undefined. Therefore, the data-replication

method [274] is used to find these hyperplanes [260, 175, 173, 166]. This method uses a single binary classifier for multi-class problems by simultaneously considering multiple binary classification problems. It starts by replicating the complete data set for each class c, adding a new dimension z with the class number, and changing each replica's target variable to a discrete one-vs-all label, discriminating c from the other classes. Next, this method learns standard binary classifiers on this new data set. As this entails multiple one-vs-all classifiers, an example is ambiguity rejected when it is predicted to be part of multiple classes.

The second category of integrated rejectors is based on Neural Networks. Typically, the models integrate the rejector and predictor into the same network architecture. For example, Geifman and El-Yaniv [191] consider a shared main network with a dedicated prediction and ambiguity rejection layer. Likewise, Vasconcelos et al. [275] extend a Neural Network with special guard neurons that use network inversion to indicate if an example is novel.

Like any other case where multiple models are available, ambiguity rejections can be performed by taking the disagreement of multiple Neural Networks that are trained to differently prioritize the classes [166, 236, 165]. Similarly as for the priory discussed agreement-based SVM-based methods, we do not consider the aforementioned works identical to ensembles as they make use of identical models, specifically combined to enable ambiguity rejection.

Both the SVM and Neural Network based learning methods require a loss function. Calibrating the loss function can significantly improve the model's reject option for two reasons. First, calibration can be used to validate that any used surrogate loss indeed results in the same optimal model as would be obtained by considering the (computationally inefficient) original loss function [276, 183, 277]. Second, calibration ensures that the obtained model with a reject option is Bayes-optimal, which enables probabilistic interpretations [278].

Rejection as a by-product In some cases, models naturally lend themselves to serve as an integrated architecture, as they require no fundamental changes to enable including a reject option.

For example, learning a one-class model for every class individually enables both ambiguity and novelty rejection. If a test example falls outside all one-class models, then it is considered novel and rejected. Similarly, it is possible to perform an ambiguity rejection for examples located in two overlapping one-class models. This technique has been applied with various one-class algorithms, for example OCSVM [178], Radial Basis Function Network (RBFN) [146] and

Droplet models [279]. Finally, when representative novel examples can be simulated, one can introduce such artificial outliers in the training set and consider these as an additional class, corresponding with novelty rejection [280, 130, 142].

4.6.4 Combining multiple rejectors

Most rejectors are tailored towards a single rejection type. However, by combining multiple rejectors one can enable both ambiguity and novelty rejection. In this section, we study how to combine the decisions of multiple rejectors.

In general, there exist two types of combination approaches, depending if the rejectors have an overlapping rejection region, meaning an example can be rejected by two or more rejectors.

First, for non-overlapping rejectors or when not interested in the example's true rejection type, the set can be combined by a logical *or*-rule: reject the example if it is rejected by any of the rejectors [281].

Second, to trace back the true rejection type in a set of overlapping rejectors, a simple *or*-rule is insufficient. For example, an example that is novel but close to the decision boundary could ideally be rejected by both a novelty rejector and an ambiguity rejector considering the proximity to the decision surface. Instead, higher-level reasoning is required, for example carefully selecting the order the rejectors are evaluated in. Typically, this is done in a multi-step architecture, stacking rejectors [282, 283] or even multiple models with a reject option [125].

4.7 Learning a model with a reject option

During the learning phase, two learning strategies to train the model with reject option exist: training the predictor and the rejector sequentially, or training both simultaneously (see Section 4.4.2). In this section, we discuss how these learning strategies are used throughout the literature and what the relevant properties are.

4.7.1 Sequential learning

Learning the predictor and rejector sequentially is typically done by first learning the predictor and then the rejector. This sequence allows the use of standard machine learning models and learning algorithms for the predictor. While the

inverse could lead to better predictors, this has not been studied in literature. Therefore, for sequential learning, we focus on algorithms to learn the rejector based on a priorly trained predictor.

We distinguish between three approaches for learning the rejector: i) optimizing a specific criterion, ii) rejecting a fixed predefined fraction of examples, and iii) taking advantage of heuristic methods.

Optimizing a specific criterion Because the rejector trades off the predictor's coverage with the predictor's performance on the non-rejected examples, learning the rejector often consists of optimizing a joint objective function where an error can be substituted by a rejection at a certain cost [211]. For classification, the cost function accounts for three costs [161]: the cost of correct prediction C_c (usually 0), the cost of misprediction C_e (usually either variable or constant equal to 1), and the cost of rejection C_r such that $C_c < C_r < C_e$. In the case of fixed costs $C_c = 0$ and $C_e = 1$, the rejection cost should be lower than a random predictor's average cost (0.5). Otherwise, the expected cost of always making a prediction would be lower than the rejection cost [195, 204]. While the rejection cost might be variable in reality [184], it is typically assumed to be constant in literature.

Under the assumption that the predictor's confidence scores are the true posterior probabilities, i.e., that the underlying examples' distribution is known, the rejector's parameters and cost can be derived through risk-minimization. This is formally known as Chow's rule [123, 206, 238, 239, 284]. Alternatively, Tortorella [196] presents an optimal reject rule based on the Receiver Operating Characteristic curve (ROC curve), which maximizes a classification utility function defined on the basis of classification and error costs. These two different yet related methods are shown to be theoretically equivalent, thanks to an important invariance property of the derivative of the ROC curve [263]. Unfortunately, using Chow's rule is not appropriate when the confidence scores are estimated from data and hence not exact. In this case, Chow's rule is no longer equivalent to the ROC approach [259] and the rejector is preferably learned following Tortorella [196]'s approach. Using Tortorella [196]'s approach, the optimal threshold is derived from the intersection between a level line, which is parametrized by the costs C_c, C_r, C_e, the TPR and False Postive Rate (FPR), and the convex hull for the ROC curve [196, 243, 121].

Conceptually, many other works follow this same approach and attempt to generalize to other settings as well as to provide solutions to edge cases. To support ensembles with different predictors, Pietraszek [264, 265] generalizes Tortorella [196]'s method to meta-classifiers, i.e., classifiers constructed from two models described by a single ROC curve. In the same vein, Dubos et al. [215]

extends Tortorella [196]'s approach to the ROC Front, which is an ensemble of diverse cost-sensitive classifiers, that proposes a variety of solutions that suit different cost-sensitive scenarios. In case of multiple classes, Cecotti and Vajda [262] still take advantage of Tortorella [196]'s method by analyzing pairwise ROC curves, and show that this can lead to a finer evaluation of the performance and thresholds definition.

In order to generate stable and smooth ROC curves, especially for small datasets, Jigang et al. [266] introduce a novel bootstrap algorithm for obtaining the class-wise sampling distributions at test time. These bootstrap distributions can be also used to estimate the posterior probabilities, improving the reject option [266]. Moreover, as altering the rejection cost C_r usually requires retraining the rejector, it is possible to extract a Pareto front [271] of the accuracy-reject curve, i.e., the curve that considers the rejection rate and the accuracy when varying the rejection cost. With this method, selecting the optimal threshold for a new rejection cost becomes fast and trivial [270].

Defining an objective function to learn the rejector is not always straightforward, as it should align with the application's needs. To tackle this challenge, some objective functions with performance constraints were proposed. For example, such function can seek for the optimal performance given a maximal allowable rejection rate [219, 241, 264, 265, 168, 182] or maximize the model's coverage given a maximum error rate [285, 264, 286, 168].

Throughout the literature, various optimization algorithms are used to find the optimal value of the objective functions mentioned before. While a complete overview of such algorithms is outside the scope of this work, we observe four types of optimizers that are used most often for sequential learning: genetic algorithms [211, 168], exhaustive search [210], iterative refinement [285], and greedy optimization [177, 146].

Rejecting a fixed fraction The rejector can be set to reject a specific proportion of the training set. Constructing such a rejector is relatively straightforward. For example, if the predictor produces confidence scores, a dependent rejector architecture can be constructed as follows. First, the training example can be rank-ordered according to the confidence measure. Second, the threshold is set such that the desired percentage of the least confidence predictions are rejected.

However, this requires a method to select the ideal fraction. There are several possible ways to approach this problem. First, the fraction of examples that should be rejected may be known based on domain knowledge about the problem [179, 234, 190, 242]. Second, it is possible to introduce a number of

adversarial examples into the training set. Then the rejection rate is set to be equal to the fraction of adversarial examples [144]. Third, one can approach the problem empirically and plot the accuracy of the model versus the rejection rate and study how the model performance increases when rejecting more examples [141, 228, 254, 172, 233]. Finally, a trivial approach is to simply reject a default percentage of examples regardless of the testing dataset [153, 192].

Heuristic methods Properly setting the parameters of the rejectors mentioned in the previous two approaches requires knowledge about the application domain. In the absence of such knowledge, one can employ heuristics that are based on the predictor's performance on the training set. For example, a rejection threshold can be set to reject any training example where the predictor makes an incorrect prediction [136]. An alternative is to allow rejecting a default fraction of the correctly predicted examples [143, 287]. In an online learning framework, Sayedi et al. [288] theoretically derive an upper bound to the number of examples that should be rejected before it is guaranteed that the misprediction rate will be limited for future predictions. Another example is given by Varshney [220] who derive theoretically an upper bound for the total cost for ensemble classification with a reject option, i.e., the sum of the misprediction and rejection costs. They show that the threshold which minimizes the theoretical cost's upper bound approximates the optimal threshold, the threshold that minimizes the empirical cost. Hence this threshold works well in practice.

When dealing with data streams, the rejection rate cannot be fixed a priori as the data distribution may change. Therefore, a dynamic and self-adapting rejection threshold can be used to decide whether or not to reject an example with a given confidence score. If the prediction is correct, the threshold is increased because the classifier is considered to be accurate. However, if the prediction is wrong, the threshold is decreased to avoid a misprediction in the future [212, 289]. The intuition of these approaches is to accept examples that, given the training data, appear easy to classify.

4.7.2 Simultaneous learning

In contrast with sequential learning, simultaneous learning (also referred to as learning **with** rejection [290]) uses a single algorithm to learn the predictor and rejector in tandem [290]. This confers the advantage of being able to exploit potential interdependencies between the predictor and rejector. The main disadvantage is that slightly changing the rejector (e.g., the costs associated with a rejection or a misprediction) requires retraining the full model [243], which may be computationally expensive. Typically, there are two strategies:

adapt an existing learner's algorithm to include the possibility of rejection or pose the task as an optimization problem, and design a suitable objective function. Next, we describe both approaches.

Adapt standard learning algorithms Sometimes the predictor's learning algorithm can be slightly adapted in order to add the reject option. For instance, one can consider rejection as an additional possible output label for the predictor and assign this target value to all the rejected training examples [146, 140, 142, 280]. However, this requires annotating examples in the training set that should be rejected.

To avoid the need to label rejections, one can resort to models that explicitly model the distribution of examples per class as this allows for a decision boundary and detect deviating examples. In Section 4.6.3, we reviewed how a combination of one-class models, one for each class, can be used for this purpose. Examples for which none of the predictors consider the example to belong to a class [194], or where an example is considered to belong to many classes [178, 245] can be rejected. Instead of learning one-class models, one can also use an ensemble of binary classifiers after performing data replication. By replicating the data and adding a dimension to indicate the class, an ensemble of one versus all binary classifiers will also allow for rejection [260, 175] (see also Section 4.6.3).

Finally, a model with a reject option can be learned by maximizing the volume under this $3D$ ROC curve [185, 202] which has on its axes, the true positive and false positive rate of the accepted examples as well as the rejection rate.

Design a rejection aware objective function A common approach to simultaneous learning is to design an objective function that captures the desiderata of the model with reject option. This is typically done by considering three costs: the cost of correct prediction C_c, the cost of rejection C_r (no prediction), and the cost of misprediction C_e. The loss for any (classification) model with reject option m, and any example x with target value y, takes the following values:

$$L(x, y) = \begin{cases} C_c & m(x) = y; \\ C_r & m(x) = \emptyset; \\ C_e & m(x) \notin \{y, \emptyset\}. \end{cases} \tag{4.6}$$

However, optimizing such a discrete loss is computationally inefficient. Hence, surrogate losses enabling computationally efficient optimization are intensively studied, for fixed costs [291, 276, 292, 293, 187, 174] and for asymmetric costs [277], where certain types of misprediction may be more costly than others.

Many theoretical properties of loss functions for rejection have been studied in the literature [277]. First, researchers study whether the convex surrogate loss is calibrated or not [293, 183]. Second, theoretical upper or lower bounds for the empirical error are investigated [291, 292, 277, 182, 192]. Third, Wegkamp and Yuan [292] analyze the effect of a penalization term in a model with reject option when learning from high-dimensional features. Fourth, researchers study how close the found solution is to the optimal one [277, 197, 130, 135]. Finally, some works focus on the convergence rate to the optimal predictor [277, 192, 292].

While most of these studies focus on binary classification, the loss functions are also applicable for other machine learning tasks. For example, multi-class classification is enabled by extending the binary cross-entropy loss [294, 184], by introducing a one-vs-all hinge loss function [129], or using the $0-1-c$ risk [183] (i.e., setting $C_c = 0$, $C_e = 1$, $C_r = c$). Additionally, for regression, a meta-loss function can be designed such that it penalizes prediction errors as well as incorrect rejections based on how large the error is [193]. Finally, Zhang and Metaxas [244] study a group-level loss function for multi-instance learning.

After designing the loss function, the problem is then solved by employing an optimization algorithm. While a full overview on optimization algorithms is outside the scope of this paper, the approaches used in learning to reject include Generalized Probabilistic Descent (GPD) [227], Stochastic Gradient Descent (SGD) [145, 247, 191, 253, 165], Sequential Minimal Optimization (SMO) algorithm for SVM models [273], Logistic Regression via Splitting and Augmented Lagrangian (LORSAL) [230], Alternating Direction Method of Multipliers (ADMM) [256], Selection with Guided Regression Risk (SGRR) based on binary search [135], Adam optimization method based on SGD [278, 184, 193], and local optimization heuristics [244, 205].

4.8 Relation between model types and rejection

Whether a model with a reject option can perform ambiguity or novelty rejection depends not only on the chosen architecture (see Section 4.6) but also on the specific machine learning model(s) that are used for the predictor and rejector. In other words, the type of rejection that can be supported depends on the kind of patterns that the model exploits to achieve its learning task. For example, a decision tree as predictor allows for ambiguity rejection as it learns discriminative patterns to separate the target classes. Because it does not capture information about the training data's distribution, novelty rejection is impossible. In this section, we discuss the different machine learning model types used in literature to enable the reject option, and study how this relates to

the available rejection types. Following Section 4.4.3, we first discuss generative models and second discriminative models.

4.8.1 Generative models

In Section 4.4.3, the generative models were divided further into two groups: the probability-based methods, in which the joint distribution is modeled, and the data-density approach, in which distance-based or similarity-based metrics are used to estimate the density of the given examples.

Probability-based methods

There is only limited literature regarding machine learning with rejection using a probability-based generative model. We distinguish between three main categories, according to the density estimator involved: i) Gaussian density estimators, ii) GMMs and iii) Neural Network encoders.

Gaussian models A rather strong assumption that can be made is that the examples of each class follow a Gaussian distribution [137]. Under this assumption, the density $P(X|Y)$ can be estimated by computing the class mean and variance through the statistical estimators and the model can be used for the predictor, rejector or both. By thresholding the class-conditional probability, novel examples can be rejected [269]. Given the estimate of $P(X|Y)$, one can also derive the data density $P(X)$ through the class prior $P(Y)$. This allows novelty rejections when comparing a test example with the estimated data density. Alternatively, the conditional posterior probability $P(Y|X)$ can be obtained through the Bayes rule. This allows ambiguity rejections. Although the Gaussian distribution is a standard choice, other parametric distributions can be used based on domain expert knowledge.

Gaussian mixture models Assuming that class-wise examples follow a single parametric distribution is often too strict. Instead, a few works in the literature take advantage of more flexible models, like GMMs [176, 235, 136], to estimate the class-conditional distribution $P(X|Y)$. Similarly to a single Gaussian distribution, approaches based on GMMs allow both novelty and ambiguity rejections. For example, Fischer et al. [176] and Laroui et al. [235] derive $P(Y|X)$ from $P(X|Y)$ through the Bayes rule, and set the rejection threshold on $P(Y|X)$ to reject ambiguous examples. Moreover, the squared distance to the closest class center is derived from $P(X|Y)$ in order to reject novelties when

the distance is too large [176]. Differently, Vailaya and Jain [136] directly set the threshold on $P(X|Y)$ for novelty rejection.

Neural Network encoder It is also possible to incorporate the estimation of Gaussian distributions as an embedded layer in a Neural Network encoder [143]. Again, a class-specific threshold on the class conditional probabilities $P(X|Y)$ is used for novelty rejection. It is unclear how ambiguity rejection is could be performed for this technique.

Modeling data density

Explicitly modeling the distribution of examples is not always possible. The distribution might be too irregular to faithfully fit with a parametric model (e.g., a GMM). An alternative approach is to empirically derive the distribution $P(X|Y)$ by counting how many training data fall nearby a given example. If this count is low, the example can be rejected as it is supposed to be novel (i.e., far from the training data). As a result, data density models are well suited as (separated) novelty rejectors. We distinguish between works that directly use the distance between examples (e.g., k-Nearest Neighbors Outlier (kNNO)), that indirectly use a distance for anomaly detection (e.g., LOF), and that use a one-class classifier (e.g., OCSVM).

Distance-based methods Measuring the distance between an example and its k-th nearest neighbor as well as counting the number of training data falling inside the sphere centered on a given example with a certain fixed radius can be used as a local density estimators. Consequently, novel examples can be rejected by setting a threshold on such a distance-based measure [254]. Alternatively, the distance to the neighbors can also be used for prediction and ambiguity rejection. For example, Arlandis et al. [254] assign to each example a class posterior probability proportional to the inverse average distance to its k nearest neighbors. Because the distance is used differently for prediction and for novelty rejection, a rejector separate from the predictor is used. Similarly, Dübuisson et al. [138] take advantage of the distance to the k-th nearest neighbor to perform novelty rejection, by setting the class-wise threshold as the maximum distance value for each class. For a detailed overview of methods that allow novelty and ambiguity rejection based on the neighbors distance, see Dalitz [127].

Anomaly detection methods Almost all unsupervised anomaly detection algorithms provide scores indicating how different an example is from the training data and, hence, can be used to estimate the data density. LOF [152] is a popular anomaly detector where high anomaly scores are obtained if the local density of an example is lower than the local density of its neighbors. LOF is not suited for ambiguity rejection and can only be used as a separated rejector [203]. An Isolation Forest is an anomaly detector that can be used as a separated novelty rejector [295]. But only if the training set contains sufficient outliers or if the algorithm is run again when new data is available [201]. This is because an Isolation Forest measures how anomalous an example is by how easily an example can be split from the observed training examples based on random splits.

One-class methods One-class classifiers learn the class border given examples that belong to the given class. For anomaly detection, there is only one class which is the normal class. One-class methods can be seen as a special case of data density modeling. In fact, one can assume that examples are uniformly distributed inside the closed decision boundary provided by the model (i.e., the hypersphere for SVM-based methods), while the probability of an example being outside it is null. As a result, instances that fall outside the decision boundary are assumed not to be generated by this distribution and can be novelty rejected.

The most common one-class model is the OCSVM [159]. Homenda et al. [149] are the first who propose to learn one OCSVM model for each class in order to reject novelties using the OCSVMs as dependent rejector. Differently, Coenen et al. [201] use a separated rejector architecture and propose a single OCSVM on the entire dataset. Wu et al. [194] argue that an optimal hypersphere is not suited for novelty rejection since instances just outside the class boundary would also be rejected. Therefore, they propose a method to enlarge the decision boundary.

4.8.2 Discriminative models

Often the dominant task is prediction and only prediction performance is considered. Hence, it is not necessary to be able to describe the distribution $P(X, Y)$ but only to derive Y from X. Discriminative models therefore estimate $P(Y|X)$ directly. A consequence is that only ambiguity rejection can be achieved using the dependent or integrated architecture, and a separated rejector is required for novelty rejection.

The vast majority of work discussing machine learning with rejection deals with discriminative models. In this section, we will give an overview of the most popular discriminative machine learning methods and discuss their ability to reject.

Decision tree and random forest classifiers are predictors which iteratively try to split the data such that finally, each leaf in the tree contains examples of just one class. Hence, the class probability in a leaf is well suited as a confidence score for ambiguity rejection [229, 180]. The model does not contain information on the data density and, hence, the model does not allow for novelty rejection. *Ensemble* classifiers often use a voting scheme to combine the output of multiple classifiers for prediction. The ensemble agreement indicates ambiguity uncertainty. This uncertainty can be used to reject examples [212, 175]. However, again there is no information on the data density available and novelty rejection is not possible. For *logistic regression* [230, 231, 232] the odds indicate the confidence and can be used for ambiguity rejection. When using an *SVM*, the distance to the decision boundary can be used to perform ambiguity rejection [244, 174].

Because discriminative prediction models do not contain information on the data density to perform novelty rejection, some authors make assumptions to relate either the model properties or the model outputs to the data density. For example, De Stefano et al. [161] argues that in a *Neural Network classifier*, an example resulting in low activities for all output neurons differs significantly from the training data. Hence, thresholding the maximal neuron activity of the output neurons makes novelty rejection possible [161]. With similar reasoning, it is argued that high neuron values in Learning Vector Quantizations (LVQs) are related to examples which are far from all learned LVQ prototypes and hence, have high epistemic uncertainty. This makes it possible, up to some level, to do both ambiguity and novelty rejection based on a discriminative model for the predictor. Therefore, this approach is often used in practice [224, 222, 252]. However, because of the assumptions which were made, it is not guaranteed that all novel examples will be rejected.

4.9 Applications of machine learning models that reject

In safety-sensitive domains, making the wrong decision can have serious consequences such as fatal accidents with self-driving cars, major breakdowns in industrial settings or incorrect diagnoses in medical applications. In these

domains, a reject option can be used to make cautious predictions. However, the number of papers discussing machine learning with rejection in practical applications is still limited. In this section, an overview of these papers is given.

Machine learning with a reject option is most frequently studied for medical applications because a wrong decision can have serious consequences. The largest group of papers handles *medical diagnosing*, the detection and classification of diseases. This detection can be automatically translated into a diagnosis if the model is sufficiently sure. Otherwise, the detection task is passed to a medical expert for verification [126]. An example is the detection of vocal pathologies using voice recording data. For this detection, a linear classifier was trained and uncertain predictions are rejected by thresholding the derived posterior probability [121]. To diagnose spine diseases based on biomechanical characteristics of the patient, the data-replication method for classification with ambiguity rejection can be used [260]. This method only predicts if two biased classifiers agree. Also cancer detection is fault-sensitive and hence, benefits from a reject option. An example is breast tumor detection using an SVM classifier with confidence-based rejection [296]. In this application, the rejection thresholds were chosen to limit the rejection rate and hence, the amount of manual effort. All these methods focus on ambiguity rejection.

Next to diagnostics applications, other biomedical applications have discovered the use of a reject option. Lotte et al. [146] perform an experiment to distinguish movement from the left or right hand based on brain activity and uses a separated novelty rejector to reject brain activity related to other activities. The rejector is trained in a supervised manner. Lewicke et al. [218] study sleep stage scoring using both types of rejection using confidence metrics derived from the neural activities of a Neural Network classifier. Another sleep stage scoring application uses a separated rejector based on LOF anomaly scores to do novelty reject, identifying patients who deviate from the training data [203]. Some papers compare the performance of multiple predictors with reject option and check which works best for the biomedical application at hand. A first paper predicts whether a diabetes drug is likely to work for a specific patient [297], a second investigates the classification of body positions based on sensors placed in the patient shoe [250]. Finally, a last medical application handles the analysis of tissue examples in which the goal is to classify every pixel of the tissues image in categories such as bone, fat or muscle [230, 231].

Besides medical applications, also applications from the field of engineering can benefit from a reject option. An example is the chemical identification of gasses [208]. In this application, time-series data is passed to two classifiers which try to classify the observed gas as soon as possible. Classification is only done if ensemble agreement is reached. Up until that moment, all predictions are rejected due to the ambiguous prediction. A similar rejection technique (i.e.,

rejecting when two classifiers disagree) is also used in an application on defect detection in software [166]. Another engineering application is fault detection in steam generators [122]. In this application, a set of one against all SVM classifiers is used and rejection is based on the distance to the decision boundary of these classifiers, allowing both rejection types.

In the domain of economics, two applications of machine learning with rejection have been proposed, both focused on ambiguity rejection and novelty rejection. The first application uses a LVQ to classify dollar bills by value [224]. Confidence metrics are obtained from the classifier for both types of rejection. The second application investigates a few rejection techniques on top of a predictor to decide whether to grant a loan [201]

Reject options are also available for the analysis of text styles [241, 298] and to read handwritten numbers, applied for ambiguity rejection [217, 134, 299, 300, 221, 254, 262, 189] as well as novelty rejection [221, 254]. All these methods use a confidence-based dependent rejector.

Finally, there is a paper which attempts to identify walkers based on their footprint [222]. This method starts by predicting or rejecting each footprint individually but afterwards, combines the information of three consecutive footprints to make a final decision.

4.10 Link to other research areas

This section briefly discusses the fields that are closely related to learning with a reject option.

4.10.1 Uncertainty quantification

The area of uncertainty quantification aims to assess how uncertain a learned model is about its predictions [301]. Traditionally, the field distinguishes between two types of uncertainties: the non-reducible aleatoric uncertainty and the reducible epistemic uncertainty [302]. The former refers to the notion of randomness in the data, such as the variability due to a non-deterministic relation between the features and the target. The latter refers to the lack of knowledge which can arise due to a small training set (data-epistemic) or incorrect model bias (bias-epistemic). To illustrate this type of uncertainty, consider trying to predict the outcome of tossing a potentially unfair coin [303]. Initially, the lack of historical data about the outcomes of flipping the coin means that we would make an incorrect prediction if asked to state the chance

that the coin will land on heads. This is situation represents high data-epistemic uncertainty. After observing the outcome of a sufficient number of coin tosses, the data-epistemic uncertainty decreases as we will be able to provide a better estimate of the chance of landing on head. However, the aleatoric (statistical) uncertainty remains due to the stochastic outcome associated with a coin flip.

These two types of uncertainty are inherently related to rejection. Ambiguity rejections are related to the aleatoric uncertainty or the bias-epistemic uncertainty, while novelty rejection follows from the data-epistemic uncertainty. However, these areas differ in their objectives. Uncertainty quantification is concerned with how to measure uncertainty. Moreover, it is important that its estimates are calibrated and in this sense meaningful convey the uncertainty. In contrast, the objective of the reject option is to abstain from making a prediction when the uncertainty is high. Therefore, it can be seen as one way to apply uncertainty quantification techniques [272, 126]. While calibrated estimates of uncertainty are not always strictly necessary in learning to reject, they can be important. For example, Chow [123] shows that correct uncertainty estimates enables finding the optimal threshold which minimizes misprediction risks.

4.10.2 Anomaly detection

Anomaly detection [304] is a canonical Data Mining task where the goal is to detect examples that do not conform to the data set's expected behavior. Anomaly detection is closely related to novelty rejection because anomalies are, by definition, rare and scarce examples and hence substantially different to what was observed in the training data [305]. In fact, novelty detection and out-of-distribution detection are subfields of anomaly detection that explicitly attempts to determine if an example is highly dissimilar to the data in the training set [74, 306, 307]. Thus anomaly detectors are often used to perform novelty rejections, typically within a separated rejector architecture.

Moreover, a reject option can be added to anomaly detectors, abstaining from processing examples for which no clear decision can be made [308]. However, enabling the reject option in an anomaly detection task is not straightforward as this is an unsupervised problem. This leads to two main challenges. First, most confidence metrics assume a supervised problem, for example measuring the distance to its decision surface. For anomaly detection, such a hard decision surface may not always exist. Instead, specialized confidence metrics are required, for example based on a quantification of an anomaly detector's model bias [308]. Second, due to the lack of labels, it is hard to learn a rejector based on standard performance metrics such as accuracy. Instead, learning to reject

in an anomaly detection setting requires unsupervised techniques, for example based on performance metrics measuring an anomaly detector's stability [309].

4.10.3 Active learning

In active learning [310, 311, 312, 313], the learning algorithm interacts with an oracle who provides feedback to help guide the learner. Its original motivation was to avoid having to label large amounts of data to achieve acceptable performance. Instead, the algorithm should identify those examples whose labels would be most beneficial to the learner. The goal is to achieve high predictive performance while avoiding the cost associated with labeling large amounts of data.

The key similarity between active learning and learning with rejection is that they both focus on examples where the model is uncertain. However, they fundamentally differ in why they are interested in this uncertainty. In active learning, the uncertainty is important at training time as its goal is to make learning more efficient with respect to the amount of labeled data that is needed to learn an accurate model. Its intuition is that if the model is uncertain about an example's label, this indicates that acquiring this label could improve the model's performance. In contrast, learning with rejection wants to capture uncertainty at test time to avoid making a misprediction. Another difference is that outliers are not of interest in active learning because knowing their label does not improve the learned model whereas novelty rejection wants to identify such examples because they may decrease the performance of the model.

However, combining these two fields could be of great use [314]. When the interaction with an oracle is possible, it may be of interest to query the rejected test examples. New data types could be identified by the novelty rejection, while ambiguity rejection may fine-tune the decision boundary.

4.10.4 Class-incremental / incomplete learning

Typically, a machine learned model assumes that all possible classes are known upfront during training. In contrast, the machine learning sub-field of class-incremental learning studies models that, during deployment, detect the occurrence of novel classes and adapt themselves to be able to correctly predict these classes.

Both novelty rejection and class-incremental learning make an open-world assumption and share the objective of detecting examples that are novel with respect to the training set. However, there are several important differences.

First, class-incremental learning is focused on identifying examples that belong to a novel class. Hence, these techniques differentiate between whether an example belongs to a novel class or is simply an outlier. In contrast, this distinction is not important from a rejection perspective. Second, class-incremental learning involves both detecting examples that belong to a novel class and retraining the model to recognize these classes.

Novelty rejection techniques can be considered in class-incremental learning. Moreover, both techniques can be combined into a single pipeline, by adapting incremental models with novelty rejected examples. For example, such examples can be used as prototypes in a kNN model.

4.11 Discussion

We have studied the subfield of machine learning with a reject option and provided a higher-level overview on existing research. To conclude, we revisit our key research questions and point to new directions that future research might take.

4.11.1 Research questions revisited

This survey is built around seven key research questions, introduced in the introduction. In this section, we revisit each of these questions and briefly summarize our findings.

How can we formalize the conditions of when a model should abstain from making a prediction? In Section 4.3, we identify two types of rejections: ambiguity rejection and novelty rejection. Ambiguity rejection abstains from making a prediction when the learned prediction model does not faithfully mimic the true underlying function in some regions of the instance space. This could be due to a non-deterministic true relation between the features and target variable, or due to a hypothesis space that is not able to capture the true relation. Novelty rejection abstains from making a prediction on examples that are too different from the training set. For such an example, there is no guarantee that the model correctly extrapolates to this untrained region, making it likely that the model mispredicts the example.

What architectures are possible for operationalizing (i.e., putting this into practice) the ability to abstain from making a prediction? We categorize

machine learning methodologies with a reject option in three different architectures, depending on the data flow at test time: separated, dependent and integrated rejector. These categories are introduced in Section 4.4.1 and mapped to existing literature in Section 4.6. Moreover, we study techniques combining a set of rejectors, enabling both ambiguity and novelty rejection in any model.

How do we learn models that have the ability to abstain from making a prediction? To learn a model with a reject option, we discuss two different learning strategies in Section 4.4.2 and related these to existing literature in Section 4.7. First, sequential learning considers learning the prediction model and the reject option as independent problems, learning one prior to the other. Second, simultaneous learning tackles both learning problems by a single algorithm, allowing to capture the interplay between the reject option and the predictor.

Which types of rejections are possible when using standard machine learning models? In Section 4.4.3, we introduce the concepts of generative and discriminative machine learning models and study their relation to ambiguity and novelty rejection, based on the information captured by each of these machine learning techniques. Moreover, in Section 4.8, we map these insights to concrete approaches presented in literature.

How can we evaluate the performance of a machine learning method with a reject option? Standard machine learning evaluation is focused towards a model's predictive quality. However, in machine learning with a reject option, there exists a trade-off between the predictive quality and number of rejected examples.

In Section 4.5, we provide an overview on techniques evaluating both the prediction and rejection quality of models with a reject option. We identify two categories: metrics evaluating models with a fixed rejection rate, and metrics giving insight into a model's prediction-rejection trade-off.

Where does the need for abstaining machine learning methods arise in real-world applications? On a high level, machine learning with a reject option is typically used in applications where incorrect decisions can have severe consequences, both financially and safety-related. These consequences motivate the need for robust and trustworthy machine learning. In Section 4.9, we provide

an overview of application areas in which machine learning with a reject option is already used.

How does machine learning with a reject option relate to other machine learning research areas? Section 4.10 shows that machine learning with a reject option is closely related to several other subfields of machine learning. In some cases, this relation leads to terminology and techniques overlapping or inspired by these other domains, while other cases show machine learning with a reject option in a broader perspective. In this survey, we related machine learning with a reject option to uncertainty quantification, anomaly detection, active learning and class-incremental learning.

Chapter 5

Machine learning with a reject option for usage profiling

This chapter is based on:
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 (ADMA2021)*, pages 1–15, Sydney, Australia, 2021. Springer Verlag

5.1 Introduction

This chapter demonstrates the applicability of machine learning with a reject option in a real-world use case, more specifically in vehicle engineering. When working on new car designs, manufacturers validate that their vehicle meets safety requirements and ensure that its components will be sufficiently durable. These evaluations involve operating the vehicle in a number of development tests, measuring several Key Performance Indicators (KPIs) with high-end sensors. However, these tests are time consuming and very costly to carry out as some sensors cost $>$ \$100 000. This precludes carrying out an extensive number of tests and hence requires selecting a small but representative subset

corresponding to typical vehicle usage. Traditionally, common real-world usage was deduced from polls and small-scale tests using high-end sensors [45, 46].

Recent developments in connected cars give manufacturers access to a rising amount of real-world vehicle usage data [48, 47, 49]. However, this data is of lower quality than the data from development tests. Unfortunately, it is impossible to directly compute many of the relevant KPIs from the data produced by the low-end sensors included in production vehicles. Traditionally, the relation between the low-end sensor data and high-end sensors KPIs is manually derived using extensive engineering insights and expertise, which is difficult and costly to scale [316, 317]. Machine learning has been proposed to overcome this challenge, by automatically inferring the relationship between the low and high-quality data [318, 319]. However, this poses a major challenge: learned models are only reliable in situations similar to their training data. In contrast, real-world vehicle usage is not limited to scenarios that did arise in the restricted number of development tests. Hence, a model's prediction for a KPI should only be trusted if it derives from a situation that was encountered during the testing phase.

We tackle this challenge by pursuing the recently emerging machine learning paradigm of incorporating a reject option into a model [128]. More specifically, we target novelty rejection, permitting a model to abstain from making a prediction for examples that substantially differ from those present in the training set. For vehicle usage analysis, using a novelty reject option provides two benefits. First, it enables analyzing real-world vehicle usage data without artifacts arising from unexpected vehicle usage. For example, manufacturers might be interested in the distributions of most common road types customers drive over. A model estimating these types might make incorrect predictions due to encountering unexpected road types. Second, a rejection indicates an unexpected vehicle usage event, which might be accounted for in future vehicle designs and vehicle development tests. For example, the rejects can indicate novel road types in the road analysis use case.

Motivated by vehicle usage analysis, we propose a framework that provides a learned model with a reject option by combining a standard predictive model with a novelty detector. We aim for separated rejector architecture, as its model-agnostic approach makes it compatible with any predictive model. This makes our approach applicable for various classification and regression tasks, which is in clear contrast with dependent and integrated rejectors that have a reject option tied to a specific class of models. For instance, such rejectors either evaluate a specific predictor's output [138, 195, 122] or directly embed the reject option into the predictor [275, 178]. Instead, a separated rejector architecture for novelty rejection simply assumes that the predictor is likely to be inaccurate on data points that are highly dissimilar to those examples in the training data.

This yields a model-agnostic approach where r models the training data using, for example, a one-class model such as a Gaussian-mixture [148] or a One-Class SVM [149, 201]. While such separated rejector architectures are typically used for classification [192, 128], we study the generalization of this architecture to regression problems.

We evaluate our approach on a novel data set we collected by monitoring a vehicle that was driven over 20 000 kilometers over the course of nearly two years. Concretely, we show how incorporating a reject option substantially improves predictive performance when estimating road roughness in the wild. To summarize, we make the following contributions:

1. Highlight the data science challenges that arise when analyzing real-world vehicle usage to inform design engineering;
2. Propose a model with a separated rejector architecture that helps overcome some of the aforementioned challenges for this task;
3. Study the separated novelty rejector architecture for regression problems;
4. Demonstrate the efficacy of our framework on a large-scale road-roughness analysis use-case.

5.2 Background on vehicle engineering: need for usage profiling

Manufacturers need to constantly trade-off performance attributes. For example, lightweight designs lead to better fuel efficiency [320], but this needs to be balanced against durability and safety aspects [321]. Instead of relying on rules of thumb for design decisions as was done in the past, pursuing environmentally friendly designs requires to align performance attributes with real-world usage.

Nowadays, both the design and its requirements can be optimized by better insights into real-world vehicle usage. Firstly, these insights help engineers to better tailor vehicle designs to the targeted customer market. For example, one could adapt vehicles for different geographical markets based on the quality of the roads within each region [44]. Secondly, validating the requirements in development tests is an expensive development phase, as it involves high-end sensors, specialized proving grounds, and skilled drivers [322]. Consequently, insights in real-world usage help to minimize testing time by selecting the most representative scenarios [323, 324].

5.3 Usage profiling: data science challenge

The vehicle usage profiling task involves deriving insights into the real-world usage of production vehicles by the customers that bought the vehicle. The goal is to improve the vehicle's design by better aligning its requirements to how the vehicle will be used in practice. Performing a proper analysis requires taking the following challenges into account:

1. **Automation** The same framework should be generic enough that it can be applied to analyze a large number of different usage parameters with minimal modifications. That is, given a new KPI, the time needed to derive a relation between this KPI and the low-end sensors should be minimized.
2. **Small data** High-end vehicle measurement data is often limited, as it is expensive to collect. The selected machine learning techniques must be able to cope with a limited amount of training data.
3. **Incomplete training data** Due to cost and time restrictions associated with testing, manufacturers can only collect ground truth data with high-end sensors for a limited number of real-life usage scenarios. The methodology should be robust to data arising from previously unseen scenarios.
4. **Identify unexpected usage** Vehicles operated in real life are used in unexpected or novel ways. It is important to automatically identify these scenarios as doing so can yield insights that can improve the designs of future vehicles.
5. **Model-agnostic architecture** Users need to fully trust the model before using it to make impactful decisions. In addition to the required robustness to novel data points, users might favor white-box techniques or want to reuse established models. To support any such model, the rejection architecture should not pose any constraints on the predictor.

5.4 Our approach for vehicle usage profiling

Our goal is to investigate how to automatically detect vehicle usage in the real world that differs from scenarios seen during the development tests in order to minimize errors arising from predictions made during unexpected usage. Formally, this translates to the following data science problem:

Given: Training data X_{train} with target Y_{train} collected in a controlled environment.
Do: Learn a model m that 1) predicts Y for examples similar to X_{train} and 2) abstains from making a prediction for examples highly dissimiliar to X_{train}.

The overall goal can be divided into three subtasks: (i) learning a predictive model capturing the relationship between the features and the target variable, (ii) learning a rejector modeling X_{train} in order to assess how novel a test example is, and (iii) integrating the rejector's output in order to decide if a prediction should be made for an example. Next, we describe each component in detail.

From the application perspective, separating the prediction and rejection tasks into different models has two main benefits. First, it allows extending existing prediction models with a reject option by simply training a rejector model. Second, this approach is agnostic to the prediction task. For example, this enables the rejector to be used in both classification as regression problems.

5.4.1 Predictor

The predictor h models the relationship $h*$ between the low-end sensors and the desired KPI. This corresponds to the standard machine learning setting, with the model learned from the vehicle development test data X_{train}. As the separated architecture is model-agnostic, it does not pose any constraints on the considered model class. Moreover, it does not require altering the underlying learning algorithm.

5.4.2 Rejector

The rejector r uses a novelty detector to model X_{train}, assessing whether an example is an outlier to the examples in X_{train} [74]. We generalize prior work on separated rejection architectures in the context of classification [148, 149], to define four criteria that a novelty detector must satisfy to be useful in the context of vehicle usage:

1. KPIs in vehicle engineering can be discrete or continuous, so the architecture must be applicable to both classification as regression problems. For regression, estimating parameters of novelty detectors such as the expected number of classes or clusters is challenging. Therefore,

the novelty detector should require little apriori knowledge about the data distributions.

2. The detector should return a novelty score instead of a hard prediction. This allows a user to control how sensitive the rejector is to novel data points. For example, this allows us to still make predictions for examples that fall just outside an One-Class Support Vector Machine (OCSVM)'s minimal sphere.

3. Novelty scores should be representative of how similar an example is to the training data. That is, the further a point is from the training set, the higher its novelty. If this were not the case, it would not be possible to tune the rejector's sensitivity by varying its threshold.

4. The model should be able to return scores falling outside the range of scores assigned to examples in the training set. If this were not the case, a novel example cannot be distinguished from the training data.

Two models satisfying these criteria for novel examples are k-Nearest Neighbors Outlier (kNNO) and the OCSVM. kNNO considers test examples that fall in low data density regions to be novel, which is a valid indicator of novelty. OCSVM has been extensively used, and it models the data by finding a hypersphere enclosing the training data.

Novelty detection typically identifies examples that are novel due to differences in the observed value of *any* feature. However, for this application, we mostly care about differences in features that are most important for the predictor. That is, we might not care if an example is novel because it arises due to the car being driven on an unusually hot or cold day. Therefore, we restrict the novelty detectors to only use the predictor's most informative features. Moreover, this is beneficial as kNNO struggles with irrelevant features and high-dimensional spaces.

5.4.3 Combined model

Given a test example x, it is necessary to integrate the rejector and predictor to arrive at a final model m. Our approach uses the following rule to determine whether to make a prediction for x:

$$m(x) = \begin{cases} \emptyset & \text{if } r(x) \leq 0 \\ h(x) & \text{otherwise.} \end{cases} \tag{5.1}$$

That is, it first applies the rejector to obtain a confidence score $r(x)$. If $r(x)$ is negative, then m abstains from making a prediction. Otherwise, it applies the

predictor h to x and returns its prediction. The rejector's confidence score is derived from the novelty detector and indicates the similarity between x and X_{train}: $r(x) = \tau - novelty_score(x|X_{train})$. The key parameter is τ, a user-defined rejection threshold which determines m's coverage ($\frac{N_{accepted}}{N}$), which is the fraction of test examples for which a prediction is made ($N_{accepted}$) over the total number of examples tested (N).

The value of τ controls the trade-off between m's predictive performance and its coverage. Lower values of τ mean that m is very conservative and will only make predictions for examples that are very similar to the training data. Hence, one would expect a higher accuracy on those examples where a prediction was made, but at the cost of abstaining from making a prediction for many test examples. In contrast, picking a larger value for τ means m will be making predictions for a greater percentage of the test examples, but it comes with an increased risk of making a larger mistake for those examples that differ from the training data.

5.5 Use-case: road-roughness analysis

We will evaluate our rejection-based machine learning methodology by analyzing real-world car usage data that we collected. Specifically, we will focus on assessing road roughness based on the inexpensive sensors that are present in most modern cars. Understanding road conditions is crucial for design as a road's roughness is one of the primary sources of vibrations, which have a large impact in terms of degrading the components in a car. We focus on the road roughness rather than the road type due to large variability of quality within different road types. For example, an ill-maintained asphalt road might be rougher than a perfectly crafted cobblestone road. Because geographic differences in exist, obtaining large-scale insights into the prevalence of different road types and their quality would enable a manufacturer to tailor a new vehicle to the expected conditions in its target market. The gold standard for assessing road roughness is the International Roughness Index (IRI) [325]. Unfortunately, the IRI must be computed using high-end laser sensors [325] which are impossible to include in production vehicles because they are too fragile and expensive. However, validating our methodology requires estimating the road-roughness for the real-world usage data. Since a laser system cannot be used for long-term measurements, we opt for a road roughness KPI derived from a temporarily installed high-end accelerometer [326]. Typically, it is only feasible to use such a sensor in development tests or a low-end end variant of luxury cars with active suspension systems. For the majority of the cars, this is not the case.

We hypothesize that

1. using the low-end sensors typically included in production vehicles will permit accurately assessing the road roughness;
2. incorporating a rejector will result in better performance than just using a predictor while still maintaining acceptable coverage; and
3. analyzing the rejected segments will give insights into novel road types encountered during daily vehicle usage.

5.5.1 Data Collection and Preprocessing

We collected two datasets using the same vehicle, a controlled data set and a real-world data set. The former contains ground truth data obtained by driving at predefined speeds over Belgium's three most common road types: smooth asphalt, rough asphalt, and concrete slabs. Additionally, we also measured the car while it was idling. Table 5.1 shows the details about the controlled data along with the observed road roughness for each road type. The real-world data set contains data arising from natural driving behavior collected over a period of 23 months. During this period, the vehicle was used for daily travel by 39 volunteer drivers for over 20 000 kilometers, mainly on Belgian roads. To anonymize the data, we did not save driver names and removed data points within a radius of 400m $+rand(100m, 200m)$ of each trip's start and end locations.

In both settings, the same data was measured. First, a high-end DC accelerometer[1] was mounted on the front wheel hub (Figure 5.1) and sampled at 2048Hz to derive the true road-roughness. To remove the low-frequent sensor drift due to the data acquisition system, we used a high-pass filter of 1Hz. Second, 16 low-end sensors were measured through the vehicle's Controller Area Network (CAN) bus with a sample-and-hold approach at 200Hz. Table 5.2 provides a full overview of the sensors and their quantities. Finally, we acquired the vehicle's location through a GPS antenna for documentation purposes.

We process the high-end accelerometer data and low-end control system data in consecutive, non-overlapping, 1-second windows. In each window, we derive the true road-roughness from the high-end accelerometer and extract the 21 statistical features listed in Table 5.3 for each of the 15 low-end sensors, which yields 315 features. Each feature is normalized using an outlier-robust scaler $(\frac{X - P_{50\%}(X_{train})}{P_{75\%}(X_{train}) - P_{25\%}(X_{train})})$, where P_X corresponds to the X'th percentile.

[1] PCB 3713B11200G



Figure 5.1: We instrumented the front wheel hub with a high-end accelerometer in order to derive the true road-roughness.

Table 5.1: The predefined driving speeds and road types the controlled data set was collected on. For each category, we indicate its size by total measurement time and indicate its 95-th percentile ($P_{95\%}$) of the true road roughness.

Scenario	**Speed (km/h)**	**Time (s)**	$P_{95\%}$**(Road roughness)**
Smooth asphalt	30, 40, ..., 100	467	0.047
Rough asphalt	30, 40, ..., 100	386	0.190
Concrete	30, 40, 50	272	0.334
Car idle	0	415	0.002

5.5.2 Experimental Methodology

The predictor is a regressor for road-roughness estimation. The rejector is a novelty detector that indicates when to abstain from predicting the roughness. Both algorithms are combined in a separated rejector architecture and trained on the data collected in the controlled setting. We evaluate performance on the controlled data using ten-fold cross-validation.

Predictor

On the controlled data set we evaluated six different regression models as shown in Table 5.4. Since this work's main focus is on the rejection component, we opt for the predictor's default parameters as selected in Scikit Learn [327]. For the real-world analysis, we focus on the two best performing models: Random Forest (RF) and XGBoost.

Table 5.2: The 15 low-end sensors used to estimate the road-roughness

Sensor	**Unit**
Vehicle speed	Km/h
Yaw	Rpm
Wheel propulsion torque	Nm
Angle steering wheel	Degrees
Tortion bar torque	Nm
Speed limit	*Unknown*
Lateral accelerations	m/s^2
Longitudinal accelerations	m/s^2
Total vehicle acceleration	m/s^2
Low-end accelerometer	*Unknown*
Gear indicator	Gear
Rotational speed front right wheel	Rpm
Rotational speed front left wheel	Rpm
Rotational speed rear right wheel	Rpm
Rotational speed rear left wheel	Rpm

Rejector

We study two types of novelty detectors that require limited a-priori knowledge about the data distributions:

1. One-class SVM (OCSVM), which is a domain-based method that attempts to model the training data's boundary. We use a RBF kernel with $\nu = 0.005$ and $\gamma = 0.1$.
2. kNNO, which is a distance-based method that attempts to capture the training set's density. We set $k = 10$ and use the Euclidean distance.

To limit the dimensionality of data, the rejector only considers the chosen predictor's 10 top-ranked features. These were determined using the Gini importance metric for RF and the average gain across all splits the feature for XGBoost.

5.5.3 Results

Figure 5.2 shows how the Mean Absolute Error (MAE) varies as a function of coverage on the real-world data. We compare each predictor/rejector model

Table 5.3: The 21 statistical features extracted from each of the low-end sensors

Feature	**Equation**
Amplitude	$max(X) - min(X)$
Crest factor	$\frac{max(\lvert X-\overline{X}\rvert)}{rms(X)}$
Extreme deviation	$max(\frac{max(X)-\overline{X}}{std(X)}, \frac{\overline{X}-min(X)}{std(X)})$
Integration	$\sum_{i=1}^{n} \frac{f(x_{i-1})+f(x_i)}{2} \Delta x_i$ with $\Delta x_i = 1$
Kurtosis	$\overline{(X-\overline{X})^4} / std(X)^4$
Markov regression	$R(1)/R(0)$ with $R(0) = \sum_{i=0}^{n-1} x_i^2$ and $R(1) = \sum_{i=0}^{n-2} (x_i x_{i+1}) + x_0 x_{n-1}$
Mean	$\overline{X} = \frac{1}{n} \sum_{i=1}^{n} x_1$
Skewness	$m_3 / m_2^{\frac{3}{2}}$ with $m_j = \overline{(X - \overline{X})^j}$
Standard deviation	$\sqrt{variance(X)}$
Sum	$\sum_{i=1}^{n} x_i$
Variance	$\overline{\lvert X - \overline{X}\rvert^2}$
Maximum	$max(X)$
Extremum	$max(\lvert X\rvert)$
Median	$\tilde{X}$
Minimum	$min(X)$
Percentile 10	$\%ile_{10\%}(X)$
Percentile 25	$\%ile_{25\%}(X)$
Percentile 75	$\%ile_{75\%}(X)$
Percentile 90	$\%ile_{90\%}(X)$
RMS	$\sqrt{\overline{X^2}}$
Average absolute deviation	$\overline{\lvert X - \overline{X}\rvert}$

Table 5.4: The performance of six different regression models on the controlled data set as determined by ten fold cross validation.

Predictor	R^2	MAE	MSE
Random Forest	0.5627	0.0258	0.0028
XGBoost	0.5435	0.0304	0.0029
Regression Tree	0.2800	0.0338	0.0046
k-Nearest Neighbors Regressor	-0.0203	0.0506	0.0065
Linear Regression	-0.1287	0.0612	0.0072
Support Vector Regressor	-0.2429	0.0717	0.0079

with a standard learned model that makes a prediction for every example. Additionally, for the predictor/rejector model, we show the MAE on the rejected examples to provide insights into the relationship between confidence scores and prediction errors.

Hypothesis 1: Low-end sensors permit accessing the road roughness. The standard model that always makes a prediction obtains a MAE of around 0.05 for both XGBoost and RF. Given that the road roughness scores lie between 0 and 0.334 in the controlled data, this performance is sufficient to distinguish among the most common road types as well as capture variation within a single road type.

Hypothesis 2: Incorporating a rejector results in better performance, while maintaining an acceptable coverage.
The results show that our proposed predictor/rejector combinations outperform the standard learned model that always makes a prediction. This holds even when they only abstain from making a prediction for a relatively small fraction of the data. In aggregate, (much) larger errors are made on the rejected examples. For example, and without making claims about the application dependent's optimal coverage, Table 5.5 shows that abstaining from making a prediction on the top 5% most novel examples reduces the model's MAE by between 10% and 15%. The MAE on the rejected examples is up to 275% higher than the MAE on the standard learned model. Our approach is also more reliable as it yields lower variances on the prediction errors. Compared to the standard model, this reduces the variance by up to 53%. The improved performance and its robustness to novel scenarios will increase users' trust and confidence in the model.

For high coverages, using the OCSVM as the novelty detector outperforms using kNNO. However, its advantage diminishes for lower coverages. This is because confidence scores based on an OCSVM are not informative when a test example

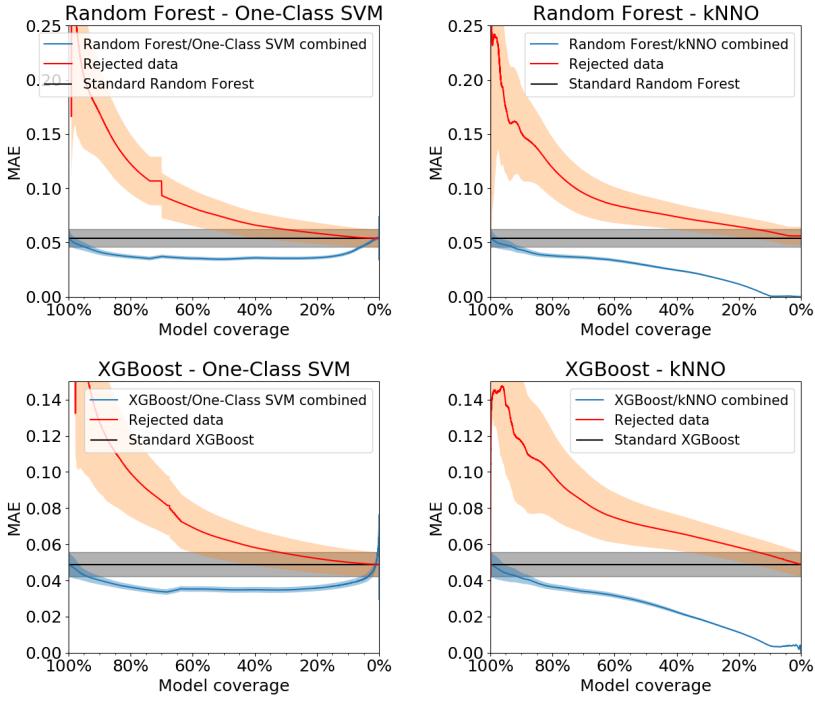


Figure 5.2: Each plot shows the MAE (solid line) and error variance (band around MAE) of a standard model that always makes a prediction (black) and a specific predictor/rejector combination (blue) as a function of coverage. The combined predictor/rejector model always outperforms the standard model. Evaluating the MAE of rejected examples (red) shows that the model makes bigger mistakes on the most novel examples, implying a relationship between the rejector's confidence score and the magnitude of the error in a prediction.

Table 5.5: The percent change in MAE and variance of the combined predictor/rejector models for 95% coverage relative to the standard model. By rejecting examples that tend to yield high prediction errors, the predictor/reject model yields both better performance and robustness to novel scenarios, thereby increasing trust.

	Predictor	**RF**		**XGBoost**	
	Rejector	**kNNO**	**OCSVM**	**kNNO**	**OCSVM**
Accepted	MAE	-11.7%	-14.5%	-9.9%	-12.0%
	Variance	-43.4%	-52.6%	-31.2%	-52.0%
Rejected	MAE	+221.7%	+275.4%	+180.7%	+228.0%

is very similar to the training data as these points fall within the hypersphere that the OCSVM uses to model the training data.

Hypothesis 3: Analyzing the rejected segments gives insights into novel road types.
A low confidence score indicates a road type that has a different relationship between the measurements from the low-end sensors and the road roughness than was observed in the training data. Indeed, inspecting the road types that receive low confidence scores reveals that these differ substantially from the road types included in the controlled data test. For example, within the top 1% least confident examples, we observe several novel road types such as those shown in Figure 5.3.

The MAE on the rejected examples shows that there is a correlation between the magnitude of the predictor's errors and the confidence scores. Larger errors are made on those examples with the lowest confidence scores. Lowering the model's coverage monotonically decreases the MAE on the rejected examples. Note that there is high variability in the MAE for coverages close to 100% as only very few examples are rejected in this region.

5.5.4 Discussion

Our approach offers the capability to abstain from estimating the roughness in novel road conditions, which has three benefits to such analysis. First, automatically detecting and declining to make predictions on examples that deviate from the controlled data yields a better-performing machine learning model which is robust to unforeseen road types. Interestingly, rejecting only a small fraction of the data already results in drastic improvements. Consequently, rejection leads to better insights about expected vehicle usage. If such insights



Figure 5.3: Analyzing the examples identified as being most novel shows that our methodology finds novel road types. Our results highlight road types such as cobblestones (left [328]), gravel roads (middle [328]), and road gutters (right [328]).

are more accurate and reliable, this results in better vehicle designs and development tests. Second, analyzing the rejected examples offers additional insights, such as identifying road types that are not currently considered in-vehicle tests. If a novel road type is frequently encountered in the real-world scenario, this indicates it could be useful to include it in future testing protocols. As we find novelty scores are correlated with the estimation errors, it is easy to prioritize analyzing novel road types based on their prevalence.

Chapter 6

General discussion

This chapter summarizes the main contributions and conclusions of this dissertation. Moreover, it discusses the limitations of our work and provides future research directions. Finally, we summarize the general lessons we have learned in this work.

6.1 Summary contributions and future directions

The goal of this dissertation was to develop techniques that help alleviate some of the challenges that practitioners encounter when applying machine learning in real-world applications. More specifically, we focused on two data-related challenges where existing methods:

1. rely on access to a sufficiently large historical database to train a model, which is often not available in practice; and
2. assume that the database is complete, that is, it contains examples of the behavior of interest that the learned model should identify.

We focused on these challenges in the context of use-cases in the application domains of condition monitoring and vehicle engineering.

In the condition monitoring use case, we proposed a framework that exploits the concept of a machine fleet to overcome the need for training data. This framework evaluates the current data arising from similarly operating machines

and has the ability to detect if a machine has failed within a fleet of similarly operating machines.

For vehicle engineering, we proposed a vehicle usage profiling approach based on machine learning with a reject option that is robust to unforeseen vehicle usage. Since traditional machine learning models' predictions are unreliable when confronted with scenarios that differ substantially from those observed in the training data (unexpected scenarios), our approach abstains from making a prediction in these cases. This has two major benefits. First, the resulting machine learning model makes fewer mistakes and is thus more reliable. Second, this methodology provides additional information by identifying unexpected scenarios that were encountered when the model was used in practice.

This dissertation presents three main contributions: our fleet-based condition monitoring framework, a survey on machine learning with a reject option, and a machine learning approach for vehicle engineering robust to unforeseen test points. These contributions and further directions are discussed in more detail below.

6.1.1 Contribution 1: Framework for fleet-based condition monitoring

Chapter 2 presents a fleet-based condition monitoring framework that **only relies on the data currently being generated**. This is in contrast with traditional approaches, which rely on historical data sets. Instead, our framework simply assumes that i) machines are simultaneously operated, ii) machines are operating in similar operational conditions, and iii) the majority of the fleet is healthy. These assumptions allow the framework to identify anomalous machines as those whose behavior deviate from the behavior observed in the majority of the machines in the fleet. It uses three inter-related generic building blocks for this identification: machine comparison, clustering, and anomaly detection. Visualizations of the outcomes of each of these blocks can provide additional insights to a domain expert.

Sections 3.3 and 3.2 demonstrated that the framework **outperforms traditional condition monitoring approaches** in an electrical machine monitoring use-case. Moreover, although our framework assumes simultaneously operating machines, Section 3.4 demonstrated that extending our framework with a short-term historical database facilitates its use for sequential end-of-production-line testing.

Although these successes in real-world use cases, several open research directions remain:

Diverse operational conditions The requirement that all machines are in a similar operational condition might be problematic in some settings. For example, the machines could be operating asynchronously or might simply perform incomparable tasks. Future research could study the former by considering a short-term historical data set, similar to the end-of-production-line testing use case. Fleet-based condition monitoring can then derive a machine's health status by comparing it to the recently acquired signatures of other machines that were operating in similar conditions. Clustering could help to identify the most similarly operating machines. A research direction to overcome the latter is to find commonalities between the machines on a component level. Instead of evaluating complete machines, the framework could analyze comparable components. To identify comparable components, ontology-based approaches such as [119, 329, 330] could be employed.

Fleet-level reasoning Operational machine data can be influenced by other machines nearby. For example, a machine's microphone can pick up noises originating from an adjacent faulty machine. In the proposed framework, both machines would be flagged as anomalous. We hypothesize that this issue can be resolved by extending the framework with a reasoning component that considers the physical topology of the fleet. In the acoustical monitoring case, the reasoning component could recognize the physical proximity of the machines and, for example, only raise an alarm for the machine with the highest anomaly score.

Scalability The framework depends on pairwise comparisons within the fleet. However, this does not scale well and limits the fleet's applicability to small fleets only. Future work could explore how to reduce the number of comparisons. A possible approach would be to explore partitioning the fleet with a computational inexpensive clustering algorithm. The number of comparisons can then be reduced by first selecting a single representative from each cluster or even the fleet as a whole [331]. Next, the comparisons are only done among these representative examples. This requires assuming similar health states within a cluster.

6.1.2 Contribution 2: Literature survey on machine learning with a reject option

Chapter 4 presents a **structured overview on machine learning with a reject option**. This machine learning subfield focuses on models that abstain from making a prediction if the model thinks the prediction is likely to be

incorrect. This results in more accurate and reliable models. Although this domain is quickly attracting interest, it currently lacks a coherent overview and suffers from non-uniformly described concepts. These shortcomings make it challenging for non-experts to adopt machine learning with a reject option in real-world applications. Our survey overcomes this by providing uniformly described terminology and concepts, as well as a structured overview of prior research in this area.

We distinguished two types of rejections and classified existing research based on three key properties: the rejection architecture, the learning perspective, and the model type. Additionally, we provided an overview of applications and link the subfield of machine learning with rejection to related fields such as uncertainty quantification, anomaly detection, and incomplete learning.

Given its significance for the usage of machine learning in real-world problems and the growing attention for robust AI, we expect machine learning with a reject option to remain an active research field. Our overview paper indicates that there are two key gaps in the literature where future research would help:

Standard settings to compare different models with a reject option A large number of machine learning algorithms with a reject option already exist. Although Section 4.6.2 presents an overview on evaluation strategies, these algorithms are typically evaluated on tailor-picked or even proprietary data sets. This makes it difficult to benchmark and compare the different approaches. Instead, we believe that standardized benchmarks would help the research community to objectively evaluate novel algorithms. Moreover, the insights generated from such a benchmark would help machine learning practitioners select the best approach for their problem of interest.

Algorithms enabling models with a reject option in domains other than classification Most papers on machine learning with a reject option study supervised classification problems. Modern machine learning tackles numerous other tasks such as regression, forecasting, and clustering, or even semi-supervised and self-supervised feedback loops. We believe that including a reject option would be beneficial for these tasks as well and hence these settings would benefit from more attention from the research community.

6.1.3 Contribution 3: Robust machine learning for vehicle engineering

Chapter 5 showcases the **benefits of machine learning with a reject option in a vehicle usage profiling application**. More specifically, we demonstrate how it facilitates affordable and large-scale vehicle usage profiling studies by robustly deriving Key Performance Indicators (KPIs) from low-end sensors.

Our methodology has the capability to abstain from making a prediction in scenarios the manufacturer did not expect. This offers two major benefits compared to traditional machine learning. First, the lack of training data for unexpected scenarios makes machine learning models inaccurate when they encounter such a case. As traditional machine learning always makes a prediction, a user cannot always trust such models. Instead, our methodology only makes a prediction in scenarios that were observed in the training set. This makes its **predictions more reliable** and enhances the user's trust in the data-driven decision-making process. Second, **identifying scenarios that were not observed in the training set** provides valuable insights: users can inspect those (rejected) scenarios in order to find trends and commonalities among them. These insights can inform the design of future vehicles and hopefully make them more aligned to the customers' expectations.

However, there are still several open research questions relating to using the proposed methodology for vehicle engineering:

Study other applications Our contribution demonstrates the applicability of machine learning with a reject option on a road roughness estimation use case. In reality, vehicle engineering needs to take several more vehicle usage aspects into account such as driving style, climate, and environmental traffic conditions. Employing our methodology for such use-cases might require additional research. For example, we only applied novelty rejection in our road-roughness analysis. Instead, ambiguity rejection might be beneficial in applications such as traffic sign recognition.

Consider temporal aspects Albeit analyzing time-series data, our vehicle usage monitoring approach analyses consecutive windows and discards any temporal aspects between them. Incorporating these aspects into the prediction and rejection components is likely to improve the analysis. For example, both the predictor and rejector can benefit from autoregressive techniques.

Improve novel scenario detection Manually studying each rejected data point for novel scenarios is time-consuming for large-scale monitoring applications. Further research is required to make this workflow more efficient. We hypothesize that clustering similar data points and only presenting each cluster's prototype significantly improves the task of novel scenario detection.

6.2 Lessons learned for data scientists

We identified several challenges during our research on the analysis of operational data. This section summarizes actionable insights and lessons learned for other data scientists working on similar applications.

6.2.1 Data does not come for free

Do not underestimate the challenges of collecting real data. The research conducted in this dissertation mainly uses self-collected data sets. We found that the data quality drastically depends on the data collection process. This process consists of several steps, such as experiment planning, instrumentation, and maintenance. All of these steps are far from trivial and need to be well-thought-out.

Be prepared Measurement campaigns are time-consuming and costly to carry out. By defining the data analysis objectives and research questions before starting data collection, a single measurement campaign can suffice for long-term research. For example, we interviewed several domain experts to define shortcomings in existing methodologies for our condition monitoring research. This allowed us to plan more than 280 scenarios which were consolidated into a test plan that was easy to follow during the measurements. The resulting data set did not just facilitate the research presented in this dissertation but can also be used to help tackle future research on topics such as diverse operational conditions and fleet-level reasoning.

Have documentation Good documentation of the measurement setup and its environment is essential for further analysis of the data set. This should include all information that is expected to be relevant for the analysis, such as sensor orientation, naming conventions, and test procedures.

In reality, written documents are rarely complete as it is hard to predict all the information that will be useful in the future. In addition, we found photos

of the setup to be extremely useful. For example, in the fleet monitoring use case, we found that some accelerometers picked up vibrations from an adjacent drive-train. However, this was not the case for all drive trains and we did not note which drive trains were physically far away from each other. Based on our photos, we could infer this information long after the actual measurement campaign.

Monitor long-term measurement setups The long-term vehicle usage setup required regular maintenance and data integrity checks. For example, these checks allowed us to quickly identify issues such as cables being bit by a marten or connection issues with the control system sensors. Interestingly, our vehicle usage methodology assigned low confidence scores to examples where the vehicle's control system produced incorrect measurements, which allowed us to identify the issue and clean the data set.

6.2.2 Domain knowledge is crucial

While it is appealing to design a fully automated analysis pipeline with machine learning, we found that it is generally better to combine automation with engineering insights.

Understand your sensors and application Understanding the sensors, the data acquisition process, and their relevance to the application is key for optimal data preprocessing. For example, in the vehicle usage monitoring use case, we found that our accelerometers suffered from signal drift. This was removed with a high-pass filter, which requires setting the cutoff frequency parameter. Being aware of the frequency band informative for road roughness analysis, we could set this parameter such that no informative data was lost.

Insights in feature selection Automated feature selection helps to speed up the machine learning workflow. However, unforeseen artifacts in the data set might mislead these feature selection algorithms. Human reasoning and critical reflection thus remain important to identify potential issues. For example, these algorithms identified the fuel level indicator sensor as being highly predictive of the road-roughness estimate. In reality, this was a quirk of the data collection process as the controlled data was collected sequentially per road type without refueling.

Data science is not always the answer Machine learning should not be applied without thought and care. In some cases, there might already exist a domain-specific solution for (a part of) the problem. Exploiting such expertise is often more time-efficient and can even lead to better results. For example, changing the data's representation based on domain expertise, visualizations, and data manipulation could make the problem relatively simple to solve. In the condition monitoring use-case, we applied this principle by using a frequency domain representation for the electrical drive-trains, which was based on signal processing expertise.

6.2.3 Can I trust you?

Models with a reject option have significant potential for use cases where trust is important. Rejection provides more context and transparency about how the learned model will behave as the user will know that the model will abstain from making predictions in novel scenarios. For example, this technology can aid autonomous driving systems by allowing them to return control to the driver by recognizing atypical or previously unencountered scenarios.

6.2.4 Divide and conquer

As a data scientist, it might be tempting to use a single machine learning model to tackle a complex problem. However, complex problems can often be subdivided into multiple sub-problems, which has two major benefits. First, simple techniques are often sufficient to tackle these sub-problems. Such techniques are typically easier to interpret, are less prone to over-fitting, and could be less "data-hungry". For example, our fleet-based condition monitoring framework consists of relatively straightforward building blocks. Tackling this problem with a single model would be far more complex. Second, the solutions to each of these sub-problems can be validated individually. Compared to end-to-end validation of a single model, this subdivision makes it easier to identify weak points. In general, this leads to easier debugging and better optimization. For example, in the vehicle engineering use-case, we separately developed and validated the predictor and the rejector. This made it relatively easy to understand the differences between using a One-Class Support Vector Machine (OCSVM) and k-Nearest Neighbors Outlier (kNNO) as the rejector. If we would have opted for an integrated rejector, a single model, this would have been far more complex.

Bibliography

[1] K. A. M. Zeinab and S. A. A. Elmustafa. Internet of Things Applications, Challenges and Related Future Technologies. *World Scientific News*, 67 (2):126–148, 2017.

[2] P. Asghari, A. M. Rahmani, and H. H. S. Javadi. Internet of Things applications: A systematic review. *Computer Networks*, 148:241–261, 2019. ISSN 13891286. doi: 10.1016/j.comnet.2018.12.008.

[3] D. J. Wile, R. Ranawaya, and Z. H. Kiss. Smart watch accelerometry for analysis and diagnosis of tremor. *Journal of Neuroscience Methods*, 230: 1–4, 2014. ISSN 1872678X. doi: 10.1016/j.jneumeth.2014.04.021.

[4] S. F. Ismail. IOE solution for a diabetic patient monitoring. *ICIT 2017 - 8th International Conference on Information Technology, Proceedings*, pages 244–248, 2017. doi: 10.1109/ICITECH.2017.8080007.

[5] M. Holt, B. Yule, D. Jackson, M. Zhu, and N. Moraveji. Ambulatory monitoring of respiratory effort using a clothing-adhered biosensor. *MeMeA 2018 - 2018 IEEE International Symposium on Medical Measurements and Applications, Proceedings*, 2018. doi: 10.1109/MeMeA.2018.8438678.

[6] D. Metcalf, S. T. Milliard, M. Gomez, and M. Schwartz. Wearables and the Internet of Things for Health. *IEEE Pulse*, 7(5):35–39, 2016. ISSN 21542287.

[7] B. H. Olstad and C. Zinner. Validation of the Polar OH1 and M600 optical heart rate sensors during front crawl swim training. *PLOS ONE*, 15(4): e0231522, apr 2020. ISSN 1932-6203. doi: 10.1371/journal.pone.0231522.

[8] M. Westerman, R. Litjens, and J.-p. Linnartz. Integration Of Probe Vehicle And Induction Loop Data: Estimation Of Travel Times And Automatic Incident Detection. Technical report, University of California, California, 1995.

[9] P. Zheng and M. Mike. An Investigation on the Manual Traffic Count Accuracy. *Procedia - Social and Behavioral Sciences*, 43:226–231, 2012. ISSN 18770428. doi: 10.1016/j.sbspro.2012.04.095.

[10] P. Händel, J. Ohlsson, M. Ohlsson, I. Skog, and E. Nygren. Smartphone-based measurement systems for road vehicle traffic monitoring and usage-based insurance. *IEEE Systems Journal*, 8(4):1238–1248, 2014. ISSN 19379234. doi: 10.1109/JSYST.2013.2292721.

[11] D. Vasisht, Z. Kapetanovic, J. ho Won, X. Jin, R. Chandra, A. Kapoor, S. N. Sinha, M. Sudarshan, and S. Stratman. Farmbeats: An IoT platform for data-driven agriculture. *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017*, pages 515–529, 2017.

[12] U. R. Mogili and B. B. V. L. Deepak. Review on Application of Drone Systems in Precision Agriculture. *Procedia Computer Science*, 133:502–509, 2018. ISSN 18770509. doi: 10.1016/j.procs.2018.07.063.

[13] S. Anandhi, R. Anitha, and V. Sureshkumar. IoT Enabled RFID Authentication and Secure Object Tracking System for Smart Logistics. *Wireless Personal Communications*, 104(2):543–560, 2019. ISSN 1572834X. doi: 10.1007/s11277-018-6033-6.

[14] A. Brewer, N. Sloan, and T. L. Landers. Intelligent tracking in manufacturing. *Journal of Intelligent Manufacturing*, 10(3):245–250, 1999. ISSN 09565515. doi: 10.1023/a:1008995707211.

[15] M.-C. Tsai. Constructing a logistics tracking system for preventing smuggling risk of transit containers. *Transportation Research Part A: Policy and Practice*, 40(6):526–536, jul 2006. ISSN 09658564. doi: 10.1016/j.tra.2005.11.001.

[16] F. Shrouf, J. Ordieres, and G. Miragliotta. Smart factories in Industry 4.0: A review of the concept and of energy management approached in production based on the Internet of Things paradigm. *IEEE International Conference on Industrial Engineering and Engineering Management*, 2015-January:697–701, 2014. ISSN 2157362X. doi: 10.1109/IEEM.2014.7058728.

[17] A. G. Frank, L. S. Dalenogare, and N. F. Ayala. Industry 4.0 technologies: Implementation patterns in manufacturing companies. *International Journal of Production Economics*, 210(January):15–26, 2019. ISSN 09255273. doi: 10.1016/j.ijpe.2019.01.004.

[18] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson. The industrial internet of things (IIoT): An analysis framework. *Computers in Industry*, 101(December 2017):1–12, 2018. ISSN 01663615. doi: 10.1016/j.compind.2018.04.015.

[19] Bundesministerium für Bildung und Forschung. Industrie 4.0, 2011.

[20] L. Farrell, T. Newman, and C. Corbel. Literacy and the workplace revolution: a social view of literate work practices in Industry 4.0. *Discourse*, 0(0):1–15, 2020. ISSN 14693739. doi: 10.1080/01596306.2020.1753016.

[21] J. Dalzochio, R. Kunst, E. Pignaton, A. Binotto, S. Sanyal, J. Favilla, and J. Barbosa. Machine learning and reasoning for predictive maintenance in Industry 4.0: Current status and challenges. *Computers in Industry*, 123:103298, 2020. ISSN 01663615. doi: 10.1016/j.compind.2020.103298.

[22] L. E. Lwakatare, A. Raj, I. Crnkovic, J. Bosch, and H. H. Olsson. Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. *Information and Software Technology*, 127 (May), 2020. ISSN 09505849. doi: 10.1016/j.infsof.2020.106368.

[23] I. Bica, A. M. Alaa, C. Lambert, and M. Schaar. From Real-World Patient Data to Individualized Treatment Effects Using Machine Learning: Current and Future Methods to Address Underlying Challenges. *Clinical Pharmacology & Therapeutics*, 109(1):87–100, jan 2021. ISSN 0009-9236. doi: 10.1002/cpt.1907.

[24] R. Caruana, P. Koch, Y. Lou, J. Gehrke, M. Sturm, and N. Elkadad. Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission. In *KDD '15 Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1721–1730, Sydney, Australia, 2015. ACM New York, NY, USA. ISBN 978-1-4503-3664-2. doi: 10.1145/2783258.2788613.

[25] T. Wuest, D. Weimer, C. Irgens, and K. D. Thoben. Machine learning in manufacturing: Advantages, challenges, and applications. *Production and Manufacturing Research*, 4(1):23–45, 2016. ISSN 21693277. doi: 10.1080/21693277.2016.1192517.

[26] J. Heaton. An empirical analysis of feature engineering for predictive modeling. *Conference Proceedings - IEEE SOUTHEASTCON*, 2016-July: 1–6, 2016. ISSN 07347502. doi: 10.1109/SECON.2016.7506650.

[27] V. Sehwag, A. N. Bhagoji, L. Song, C. Sitawarin, D. Cullina, M. Chiang, and P. Mittal. Analyzing the Robustness of Open-World Machine Learning.

In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security - AISec'19*, pages 105–116, New York, New York, USA, 2019. ACM Press. ISBN 9781450368339. doi: 10.1145/3338501.3357372.

[28] B. K. N. Rao. *Handbook of Condition Monitoring*. Springer Netherlands, Dordrecht, 1998. ISBN 978-94-010-6065-3. doi: 10.1007/978-94-011-4924-2.

[29] K. Maes, L. Van Meerbeeck, E. Reynders, and G. Lombaert. Validation of vibration-based structural health monitoring on retrofitted railway bridge KW51. *Mechanical Systems and Signal Processing*, 165(July 2021):108380, feb 2022. ISSN 08883270. doi: 10.1016/j.ymssp.2021.108380.

[30] B. E. Yousfi, A. Soualhi, K. Medjaher, and F. Guillet. Electromechanical modeling of a motor–gearbox system for local gear tooth faults detection. *Mechanical Systems and Signal Processing*, 166(September 2021):108435, mar 2022. ISSN 08883270. doi: 10.1016/j.ymssp.2021.108435.

[31] C. R. C. Farrar, H. Sohn, and K. Worden. Data Normalization : A Key For Structural Health Monitoring, 2001.

[32] K. Gryllias and I. Antoniadis. A Support Vector Machine approach based on physical model training for rolling element bearing fault detection in industrial environments. *Engineering Applications of Artificial Intelligence*, 25(2):326–344, mar 2012. ISSN 09521976. doi: 10.1016/j.engappai.2011.09.010.

[33] H. Tsunashima. Condition Monitoring of Railway Tracks from Car-Body Vibration Using a Machine Learning Technique. *Applied Sciences*, 9(13): 2734, jul 2019. ISSN 2076-3417. doi: 10.3390/app9132734.

[34] C. Yiakopoulos, K. Gryllias, and I. Antoniadis. Rolling element bearing fault detection in industrial environments based on a K-means clustering approach. *Expert Systems with Applications*, 38(3):2888–2911, mar 2011. ISSN 09574174. doi: 10.1016/j.eswa.2010.08.083.

[35] R. N. Toma, A. E. Prosvirin, and J.-M. Kim. Bearing Fault Diagnosis of Induction Motors Using a Genetic Algorithm and Machine Learning Classifiers. *Sensors*, 20(7):1884, mar 2020. ISSN 1424-8220. doi: 10.3390/s20071884.

[36] F. Zorriassatine, A. Al-Habaibeh, R. M. Parkin, M. R. Jackson, and J. Coy. Novelty detection for practical pattern recognition in condition monitoring of multivariate processes: A case study. *International Journal of Advanced Manufacturing Technology*, 25(9-10):954–963, 2005. ISSN 02683768. doi: 10.1007/s00170-004-2174-8.

[37] A. Stetco, F. Dinmohammadi, X. Zhao, V. Robu, D. Flynn, M. Barnes, J. Keane, and G. Nenadic. Machine learning methods for wind turbine condition monitoring: A review. *Renewable Energy*, 133:620–635, 2019. ISSN 18790682. doi: 10.1016/j.renene.2018.10.047.

[38] C. Yang, J. Liu, Y. Zeng, and G. Xie. Real-time condition monitoring and fault detection of components based on machine-learning reconstruction model. *Renewable Energy*, 133:433–441, 2019. ISSN 18790682. doi: 10.1016/j.renene.2018.10.062.

[39] L. Guo, Y. Yu, A. Duan, H. Gao, and J. Zhang. An unsupervised feature learning based health indicator construction method for performance assessment of machines. *Mechanical Systems and Signal Processing*, 167 (PB):108573, 2022. ISSN 10961216. doi: 10.1016/j.ymssp.2021.108573.

[40] J. Lee, H.-A. Kao, and S. Yang. Service Innovation and Smart Analytics for Industry 4.0 and Big Data Environment. In *6th Conference on Industrial Product-Service Systems*, volume 16, pages 3–8, Windsor, Ontario, Canada, 2014. Elsevier B.V. ISBN 2212-8271. doi: 10.1016/j.procir.2014.02.001.

[41] V. Zaccaria, A. D. Fentaye, M. Stenfelt, and K. G. Kyprianidis. Probabilistic Model for Aero-Engines Fleet Condition Monitoring. *Aerospace*, 7(6):66, may 2020. ISSN 2226-4310. doi: 10.3390/aerospace7060066.

[42] T. Olsson, E. Ramentol, M. Rahman, M. Oostveen, and K. Kyprianidis. A data-driven approach for predicting long-term degradation of a fleet of micro gas turbines. *Energy and AI*, 4:100064, jun 2021. ISSN 26665468. doi: 10.1016/j.egyai.2021.100064.

[43] M. Verbeke, A. Murgia, T. Tourwé, and E. Tsiporkova. Fleet-Based Remaining Useful Life Prediction of Safety-critical Electronic Devices. In *Advances in Reliability, Risk and Safety Analysis with Big Data: Proceedings of the 57th ESReDA Seminar*, pages 23–24, 2019. ISBN 978-92-76-23807-2. doi: 10.2760/836400.

[44] J. Gnamm, J. Lundgren, K. Stricker, and M. Nilvall. Winning in Europe: Truck strategies for the next decade, sep 2012.

[45] R. G. Willén, M. Groβkopf, S. Streicher, R. Allouch, and T. M. Heim. Fokus Internationalisierung: Herausforderungen in der Definition der Lastdaten für Nutzfahrzeuge. In *Tagung des DVM-Arbeitskreises Betriebsfestigkeit*, Ingolstadt, Germany, 2014.

[46] H. Winterling, R. Teutsch, and C. W. R. Müller. Ableitung Einsatzspezifischer Fahrprogramme für die Betriebsfestigkeitserprobung

schwerer Nutzfahrzeuge – Beispiel Baustellensegment. In *41. Tagung des DVM- Arbeitskreises Betriebsfestigkeit*, Ingolstadt, Germany, 2014.

[47] Markets and Markets. Connected Car Market by Service (OTA Update, Navigation, Cybersecurity, Multimedia Streaming, Social Media, e-Call, Autopilot, Home Integration and Other), Form, End Market (OE, Aftermarket), Network, Transponder, Hardware and Region - Global Forecast to. Technical report, Markets and Markets, 2020.

[48] A. F. A. Hamid, M. T. A. Rahman, S. F. Khan, A. H. Adom, M. A. Rahim, N. A. Rahim, M. H. N. Ismail, and A. Norizan. Connected car: Engines diagnostic via Internet of Things (IoT). *Journal of Physics: Conference Series*, 908(1):012079, oct 2017. ISSN 1742-6588. doi: 10.1088/1742-6596/908/1/012079.

[49] K. Stricker, R. Wegener, and M. Anding. Big Data revolutioniert die Automobilindustrie, 2014.

[50] P. Navarro, C. Fernández, R. Borraz, and D. Alonso. A Machine Learning Approach to Pedestrian Detection for Autonomous Vehicles Using High-Definition 3D Range Data. *Sensors*, 17(12):18, dec 2016. ISSN 1424-8220. doi: 10.3390/s17010018.

[51] S. Wickramanayake and D. H. Bandara. Fuel consumption prediction of fleet vehicles using Machine Learning: A comparative study. *2nd International Moratuwa Engineering Research Conference, MERCon 2016*, pages 90–95, 2016. doi: 10.1109/MERCon.2016.7480121.

[52] J. Park, Z. Chen, L. Kiliaris, M. L. Kuang, M. A. Masrur, A. M. Phillips, and Y. L. Murphey. Intelligent vehicle power control based on machine learning of optimal control parameters and prediction of road type and traffic congestion. *IEEE Transactions on Vehicular Technology*, 58(9): 4741–4756, 2009. ISSN 00189545. doi: 10.1109/TVT.2009.2027710.

[53] K. Hendrickx, W. Meert, Y. Mollet, J. Gyselinck, B. Cornelis, K. Gryllias, and J. Davis. A general anomaly detection framework for fleet-based condition monitoring of machines. *Mechanical Systems and Signal Processing*, 139:106585, may 2020. ISSN 08883270. doi: 10.1016/j.ymssp.2019.106585.

[54] D. Siegel. *Prognostics and Health Assessment of a Multi-Regime System using a Residual Clustering Health Monitoring Approach*. PhD thesis, University of Cincinnati, 2013.

[55] F. Meinguet, E. Semail, X. Kestelyn, Y. Mollet, and J. Gyselinck. Change-detection algorithm for short-circuit fault detection in closed-loop AC

drives. *IET Electric Power Applications*, 8(5):165–177, may 2014. ISSN 1751-8660. doi: 10.1049/iet-epa.2012.0316.

[56] International Organization for Standardization. ISO 10816-3:2009 – Mechanical vibration – Evaluation of machine vibration by measurements on non-rotating parts – Part 3: Industrial machines with nominal power above 15 kW and nominal speeds between 120 r/min and 15 000 r/min when measured in situ, 2009.

[57] X. Jia, C. Jin, M. Buzza, Y. Di, D. Siegel, and J. Lee. A deviation based assessment methodology for multiple machine health patterns classification and fault detection. *Mechanical Systems and Signal Processing*, 99(June): 244–261, jan 2018. ISSN 08883270. doi: 10.1016/j.ymssp.2017.06.015.

[58] J. Lee, F. Wu, W. Zhao, M. Ghaffari, L. Liao, and D. Siegel. Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications. *Mechanical Systems and Signal Processing*, 42(1-2):314–334, jan 2014. ISSN 08883270. doi: 10.1016/j.ymssp.2013.06.004.

[59] R. Liu, B. Yang, E. Zio, and X. Chen. Artificial intelligence for fault diagnosis of rotating machinery: A review. *Mechanical Systems and Signal Processing*, 108:33–47, aug 2018. ISSN 08883270. doi: 10.1016/j.ymssp.2018.02.016.

[60] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115:213–237, jan 2019. ISSN 08883270. doi: 10.1016/j.ymssp.2018.05.050.

[61] B. Yang, T. Han, and J. An. ART–KOHONEN neural network for fault diagnosis of rotating machinery. *Mechanical Systems and Signal Processing*, 18(3):645–657, may 2004. ISSN 08883270. doi: 10.1016/S0888-3270(03)00073-6.

[62] M. Unal, M. Onat, M. Demetgul, and H. Kucuk. Fault diagnosis of rolling bearings using a genetic algorithm optimized neural network. *Measurement*, 58:187–196, dec 2014. ISSN 02632241. doi: 10.1016/j.measurement.2014.08.041.

[63] M. Schlechtingen and I. Ferreira Santos. Comparative analysis of neural network and regression based condition monitoring approaches for wind turbine fault detection. *Mechanical Systems and Signal Processing*, 25(5): 1849–1875, jul 2011. ISSN 08883270. doi: 10.1016/j.ymssp.2010.12.007.

[64] F. Jia, Y. Lei, J. Lin, X. Zhou, and N. Lu. Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. *Mechanical Systems and Signal Processing*, 72-73:303–315, 2016. ISSN 10961216. doi: 10.1016/j.ymssp.2015.10.025.

[65] S. Khan and T. Yairi. A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, 107:241–265, 2018. ISSN 10961216. doi: 10.1016/j.ymssp.2017.11.024.

[66] Z. Chen, K. Gryllias, and W. Li. Intelligent Fault Diagnosis for Rotary Machinery Using Transferable Convolutional Neural Network. *IEEE Transactions on Industrial Informatics*, pages 1–1, 2019. ISSN 1551-3203. doi: 10.1109/TII.2019.2917233.

[67] M. Ge, R. Du, G. Zhang, and Y. Xu. Fault diagnosis using support vector machine with an application in sheet metal stamping operations. *Mechanical Systems and Signal Processing*, 18(1):143–159, jan 2004. ISSN 08883270. doi: 10.1016/S0888-3270(03)00071-2.

[68] A. Widodo and B.-S. Yang. Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical Systems and Signal Processing*, 21(6):2560–2574, aug 2007. ISSN 08883270. doi: 10.1016/j.ymssp.2006.12.007.

[69] A. Mauricio, C. Freitas, J. Cuenca, K. Gryllias, B. Cornelis, K. Janssens, and K. Hendrickx. Condition monitoring of gears under medium rotational speed. In *24th Internation Congress on Sound and Vibration*, page 8, London, 2017.

[70] L. Bull, K. Worden, G. Manson, and N. Dervilis. Active learning for semi-supervised structural health monitoring. *Journal of Sound and Vibration*, 437:373–388, dec 2018. ISSN 0022460X. doi: 10.1016/j.jsv.2018.08.040.

[71] T. Rogers, K. Worden, R. Fuentes, N. Dervilis, U. Tygesen, and E. Cross. A Bayesian non-parametric clustering approach for semi-supervised Structural Health Monitoring. *Mechanical Systems and Signal Processing*, 119:100–119, mar 2019. ISSN 08883270. doi: 10.1016/j.ymssp.2018.09.013.

[72] L. Zhang, Q. Cao, and J. Lee. Performance Assessment for a Fleet of Machines Using a Combined Method of Ant-Based Clustering and CMAC. *Advances in Mechanical Engineering*, 5:603071, jan 2013. ISSN 1687-8140. doi: 10.1155/2013/603071.

[73] C. Sobie, C. Freitas, and M. Nicolai. Simulation-driven machine learning: Bearing fault classification. *Mechanical Systems and Signal Processing*, 99:403–419, 2018. ISSN 10961216. doi: 10.1016/j.ymssp.2017.06.025.

[74] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, jun 2014. ISSN 01651684. doi: 10.1016/j.sigpro.2013.12.026.

[75] K. Gryllias, I. Antoniadis, and C. Yiakopoulos. A novel semi-supervised mathematical morphology-based fault detection & classification method for rolling element bearings. In *The 22nd International Congress on Sound and Vibration*, page 8, Florence, 2015.

[76] Z. H. Abu-El-Zeet and V. C. Patel. METHOD OF CONDITION MONITORING (Patent US 7275018 B2), 2006.

[77] G. West, S. McArthur, and D. Towle. Industrial implementation of intelligent system techniques for nuclear power plant condition monitoring. *Expert Systems with Applications*, 39(8):7432–7440, jun 2012. ISSN 09574174. doi: 10.1016/j.eswa.2012.01.107.

[78] P. Li, J. Eickmeyer, and O. Niggemann. Data Driven Condition Monitoring of Wind Power Plants Using Cluster Analysis. In *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pages 131–136. IEEE, sep 2015. ISBN 978-1-4673-9200-6. doi: 10.1109/CyberC.2015.16.

[79] S. Al-Dahidi, F. Di Maio, P. Baraldi, E. Zio, and R. Seraoui. A framework for reconciliating data clusters from a fleet of nuclear power plants turbines for fault diagnosis. *Applied Soft Computing*, 69:213–231, aug 2018. ISSN 15684946. doi: 10.1016/j.asoc.2018.04.044.

[80] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58, jul 2009. ISSN 03600300. doi: 10.1145/1541880.1541882.

[81] N. H. Pontoppidan, S. Sigurdsson, and J. Larsen. Condition monitoring with mean field independent components analysis. *Mechanical Systems and Signal Processing*, 19(6):1337–1347, nov 2005. ISSN 08883270. doi: 10.1016/j.ymssp.2005.07.005.

[82] A. Purarjomandlangrudi, A. H. Ghapanchi, and M. Esmalifalak. A data mining approach for fault diagnosis: An application of anomaly detection algorithm. *Measurement*, 55:343–352, sep 2014. ISSN 02632241. doi: 10.1016/j.measurement.2014.05.029.

[83] V. Vercruyssen, W. Meert, G. Verbruggen, K. Maes, R. Baumer, and J. Davis. Semi-Supervised Anomaly Detection with an Application to Water Analytics. In *2018 IEEE International Conference on Data Mining (ICDM)*, volume 2018-Novem, pages 527–536. IEEE, nov 2018. ISBN 978-1-5386-9159-5. doi: 10.1109/ICDM.2018.00068.

[84] M. Wong, L. Jack, and A. Nandi. Modified self-organising map for automated novelty detection applied to vibration signal monitoring. *Mechanical Systems and Signal Processing*, 20(3):593–610, apr 2006. ISSN 08883270. doi: 10.1016/j.ymssp.2005.01.008.

[85] B. L. Matthews, S. Das, K. Bhaduri, K. Das, R. Martin, and N. Oza. Discovering Anomalous Aviation Safety Events Using Scalable Data Mining Algorithms. *Journal of Aerospace Information Systems*, 10(10): 467–475, 2013. ISSN 2327-3097. doi: 10.2514/1.I010080.

[86] W. Jacobs, H. Edwards, V. Kadirkamanathan, and A. R. Mills. Inter-engine variation analysis for health monitoring of aerospace gas turbine engines. *European PHM*, 2018.

[87] V. J. Hodge, S. O'Keefe, M. Weeks, and A. Moulds. Wireless Sensor Networks for Condition Monitoring in the Railway Industry: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1088–1106, jun 2015. ISSN 1524-9050. doi: 10.1109/TITS.2014.2366512.

[88] E. Fumeo, L. Oneto, and D. Anguita. Condition Based Maintenance in Railway Transportation Systems Based on Big Data Streaming Analysis. *Procedia Computer Science*, 53(1):437–446, 2015. ISSN 18770509. doi: 10.1016/j.procs.2015.07.321.

[89] K. Hendrickx, W. Meert, B. Cornelis, K. Gryllias, and J. Davis. Similarity-based anomaly score for fleet-based condition monitoring. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society, PHM*. PHM Society, 2020.

[90] K. Hendrickx, W. Meert, J. P. D. C. Patricio, B. Cornelis, K. Gryllias, and J. Davis. Fleet-based health monitoring for end-of-production-line and operational testing. In *Online Proceedings of ISMA2020*, Leuven, Belgium, 2020. KU Leuven.

[91] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, mar 2009. ISBN 0-13-604259-7. doi: 10.1017/S0269888900007724.

[92] C. C. Aggarwal and S. Sathe. *Outlier Ensembles*. Springer International Publishing, Cham, 2017. ISBN 978-3-319-54764-0. doi: 10.1007/978-3-319-54765-7.

[93] R. B. Randall. *Vibration-based Condition Monitoring*. John Wiley & Sons, Ltd, Chichester, UK, jan 2011. ISBN 9780470977668. doi: 10.1002/9780470977668.

[94] M. Müller. Dynamic Time Warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer-Verlag Berlin Heidelberg, 1 edition, 2007. ISBN 978-3-540-74047-6.

[95] D. J. Bemdt and J. Clifford. Using dynamic time warping to find patterns in time series. In *AAAIWS'94 Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 359–370, Seattle, 1994. AAAI Press.

[96] D. F. Silva, G. E. A. P. A. Batista, and E. Keogh. On the Effect of Endpoints on Dynamic Time Warping. In *SIGKDD MiLeTS'16*, page 10, San Francisco, California, 2016. ACM.

[97] K. Hendrickx, W. Meert, B. Cornelis, K. Janssens, K. Gryllias, and J. Davis. A Fleet-Wide Approach for Condition Monitoring of Similar Machines Using Time-Series Clustering. In Springer, editor, *Proceedings of the 6th International Conference on Condition Monitoring of Machinery in Non-Stationary Operations, CMMNO'2018*, pages 101–110. Springer, Cham, Santander, Spain, jun 2019. doi: 10.1007/978-3-030-11220-2_11.

[98] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, 1990. ISBN 0471735787.

[99] H. S. Park and C. H. Jun. A simple and fast algorithm for K-medoids clustering. *Expert Systems with Applications*, 36(2 PART 2):3336–3341, 2009. ISSN 09574174. doi: 10.1016/j.eswa.2008.01.039.

[100] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, dec 2007. ISSN 0960-3174. doi: 10.1007/s11222-007-9033-z.

[101] F. Murtagh. A Survey of Recent Advances in Hierarchical Clustering Algorithms. *The Computer Journal*, 26(4):354–359, nov 1983. ISSN 0010-4620. doi: 10.1093/comjnl/26.4.354.

[102] M. Ester, H.-P. Kriegel, S. Jorg, and X. Xu. A Density-Based Clustering Algorithms for Discovering Clusters. In *KDD'96 Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon, 1996.

[103] J. H. Ward. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301):236–244, mar 1963. ISSN 0162-1459. doi: 10.1080/01621459.1963.10500845.

[104] V. P. Lessig. Comparing Cluster Analyses with Cophenetic Correlation. *Journal of Marketing Research*, 9(1):82, feb 1972. ISSN 00222437. doi: 10.2307/3149615.

[105] R. Sharifi and M. Ebrahimi. Detection of stator winding faults in induction motors using three-phase current monitoring. *ISA Transactions*, 50(1): 14–20, 2011. ISSN 00190578. doi: 10.1016/j.isatra.2010.10.008.

[106] Y. Mollet, M. Pergolesi, M. Sarrazin, K. Janssens, H. Van der Auweraer, P. Chiariotti, P. Castellini, and J. Gyselinck. Multi-Physical Signature Analysis of Induction Machines under Unbalanced Supply Voltage. In *2018 XIII International Conference on Electrical Machines (ICEM)*, volume 32, pages 2378–2384, Alexandroupoli, Greece, sep 2018. IEEE. ISBN 978-1-5386-2477-7. doi: 10.1109/ICELMACH.2018.8506840.

[107] Sang-Joon Lee, Jun-Koo Kang, and Seung-Ki Sul. A new phase detecting method for power conversion systems considering distorted conditions in power system. In *Conference Record of the 1999 IEEE Industry Applications Conference. Thirty-Forth IAS Annual Meeting (Cat. No.99CH36370)*, volume 4, pages 2167–2172, Phoenix, AZ, USA, 1999. IEEE. ISBN 0-7803-5589-X. doi: 10.1109/IAS.1999.798754.

[108] J. H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77, 1997. ISSN 13845810. doi: 10.1023/A:1009778005914.

[109] G. Maruthi and K. Panduranga Vittal. Electrical Fault Detection in Three Phase Squirrel Cage Induction Motor by Vibration Analysis using MEMS Accelerometer. In *2005 International Conference on Power Electronics and Drives Systems*, volume 2, pages 838–843. IEEE, 2006. ISBN 0-7803-9296-5. doi: 10.1109/PEDS.2005.1619804.

[110] P. Tavner. Review of condition monitoring of rotating electrical machines. *IET Electric Power Applications*, 2(4):215, 2008. ISSN 17518660. doi: 10.1049/iet-epa:20070280.

[111] R. Randall, Z. Y. Chin, W. Smith, and P. Borghesani. Measurement and use of transmission error for diagnostics of gears. In *Surveillance, Vishno and AVE conferences*, Lyon, France, 2019.

[112] A. Palermo, L. Britte, K. Janssens, D. Mundo, and W. Desmet. The measurement of Gear Transmission Error as an NVH indicator: Theoretical discussion and industrial application via low-cost digital encoders to an all-electric vehicle gearbox. *Mechanical Systems and Signal Processing*, 110:368–389, 2018. ISSN 10961216. doi: 10.1016/j.ymssp.2018.03.005.

[113] A. Palermo, J. Anthonis, D. Mundo, and W. Desmet. A novel gear test rig with adjustable shaft compliance and misalignments part I: design.

In *Advances in Condition Monitoring of Machinery in Non-Stationary Operations*, pages 497–506. Springer, Ferrara, Italy, 2013.

[114] A. Palermo, J. Anthonis, D. Mundo, and W. Desmet. A novel gear test rig with adjustable shaft compliance and misalignments. Part II: Instrumentation. In *Advances in Condition Monitoring of Machinery in Non-Stationary Operations*, pages 507–515. Springer, Ferrara, Italy, 2013.

[115] T. Tamarozzi, P. Jiranek, D. Park, M. Vivet, S. Donders, and S. Flock. Democratization of simulation in the design engineering of gear boxes. In *Proceedings of NAFEMS Conference: Simulation Based Engineering*, Neuendettelsa, Germany, 2017.

[116] A. Palermo, D. Mundo, and W. Desmet. *Development of advanced multibody models for gear dynamics*. PhD thesis, Università della Calabria, 2012.

[117] F. Shang, Y. Gan, and Y. Guo. Hexahedral mesh generation via constrained quadrilateralization. *PLOS ONE*, 12(5):e0177603, may 2017. ISSN 1932-6203. doi: 10.1371/journal.pone.0177603.

[118] G. Medina-Oliva, A. Voisin, M. Monnin, F. Peysson, and J.-B. Léger. Prognostics assessment using fleet-wide ontology. In *Annual Conference of the Prognostics and Health Management Society 2012, PHM Conference 2012*, Minneapolis, Minnesota, 2012. ISBN 9781936263059.

[119] G. Medina-Oliva, A. Voisin, M. Monnin, and J. B. Léger. Predictive diagnosis based on a fleet-wide ontology approach. *Knowledge-Based Systems*, 68:40–57, 2014. ISSN 09507051. doi: 10.1016/j.knosys.2013.12.020.

[120] A. Voisin, G. Medina-Oliva, M. Monnin, J.-B. Léger, and B. Iung. Fleet-wide Diagnostic and Prognostic Assessment. In *Annual Conference of the Prognostics and Health Management Society*, volume 3, 2011.

[121] C. Kotropoulos and G. R. Arce. Linear classifier with reject option for the detection of vocal fold paralysis and vocal fold edema. *Eurasip Journal on Advances in Signal Processing*, 2009, 2009. ISSN 16876172. doi: 10.1155/2009/203790.

[122] C. Zou, E.-h. Zheng, H.-w. Xu, and L. Chen. Cost-sensitive Multi-class SVM with Reject Option: A Method for Steam Turbine Generator Fault Diagnosis. *International Journal of Computer Theory and Engineering*, 3 (1):77–83, 2011. ISSN 17938201. doi: 10.7763/IJCTE.2011.V3.286.

[123] C. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, jan 1970. ISSN 0018-9448. doi: 10.1109/TIT.1970.1054406.

[124] M. E. Hellman. The Nearest Neighbor Classification Rule with a Reject Option. *IEEE Transactions on Systems Science and Cybernetics*, 6(3): 179–185, 1970. ISSN 21682887. doi: 10.1109/TSSC.1970.300339.

[125] P. Pudil, J. Novovičova, S. Bláha, and J. Kittler. Multistage pattern recognition with reject option. In *Proceedings - International Conference on Pattern Recognition*, volume 2, pages 92–95, 1992. ISBN 0818629150. doi: 10.1109/ICPR.1992.201729.

[126] B. Kompa, J. Snoek, and A. L. Beam. Second opinion needed: communicating uncertainty in medical machine learning. *npj Digital Medicine*, 4(1):4, dec 2021. ISSN 2398-6352. doi: 10.1038/s41746-020-00367-3.

[127] C. Dalitz. Reject options and confidence measures for kNN classifiers. *Schriftenreihe des Fachbereichs Elektrotechnik und Informatik Hochschule Niederrhein*, 8(2009):16–38, 2009.

[128] K. Hendrickx, L. Perini, D. Van der Plas, W. Meert, and J. Davis. Machine Learning with a Reject Option: A survey. 2021.

[129] H. G. Ramaswamy, A. Tewari, and S. Agarwal. Consistent algorithms for multiclass classification with an abstain option. *Electronic Journal of Statistics*, 12(1):530–554, 2018. ISSN 1935-7524. doi: 10.1214/17-EJS1388.

[130] L. Ziyin, Z. T. Wang, P. P. Liang, R. Salakhutdinov, L. P. Morency, and M. Ueda. Deep gamblers: Learning to abstain with portfolio theory. *Advances in Neural Information Processing Systems*, 32(NeurIPS):1–17, 2019. ISSN 10495258.

[131] I. Flores. An Optimum Character Recognition System Using Decision Functions. *IRE Transactions on Electronic Computers*, EC-7(2):180, 1958. ISSN 03679950. doi: 10.1109/TEC.1958.5222530.

[132] K. Fukunaga and D. Kessell. Application of optimum error-reject functions. *IEEE Transactions on Information Theory*, 18(6):814–817, nov 1972. ISSN 0018-9448. doi: 10.1109/TIT.1972.1054919.

[133] L. Gyorfi, Z. Gyorfi, and I. Vajda. Bayesian Decision With Rejection. *Problems of control and information theory*, 8(5-6):445–452, 1979. ISSN 03702529.

[134] L. Xu, A. Krzyzak, C. Y. C. Suen, A. Krzyżak, and C. Y. C. Suen. Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22 (3):418–435, 1992. ISSN 00189472. doi: 10.1109/21.155943.

[135] W. Jiang, Y. Zhao, and Z. Wang. Risk-Controlled Selective Prediction for Regression Deep Neural Network Models. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, jul 2020. ISBN 978-1-7281-6926-2. doi: 10.1109/IJCNN48605.2020.9207676.

[136] A. Vailaya and A. Jain. Reject option for VQ-based Bayesian classification. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 2, pages 48–51. IEEE Comput. Soc, 2000. ISBN 0-7695-0750-6. doi: 10.1109/ICPR.2000.906016.

[137] B. Dubuisson and M. Masson. A statistical decision rule with incomplete knowledge about classes. *Pattern Recognition*, 26(1):155–165, jan 1993. ISSN 00313203. doi: 10.1016/0031-3203(93)90097-G.

[138] B. Dübuisson, M. Usai, and P. Malvache. Computer aided system diagnostic with an incomplete learning set. *Progress in Nuclear Energy*, 15 (C):875–880, jan 1985. ISSN 01491970. doi: 10.1016/0149-1970(85)90121-0.

[139] L. Cordella, C. De Stefano, C. Sansone, and M. Vento. An adaptive reject option for LVQ classifiers. In *Lecture Notes in Computer Science*, volume 974, pages 68–73. Springer Berlin Heidelberg, 1995. ISBN 3540602984. doi: 10.1007/3-540-60298-4_238.

[140] G. Vasconcelos, M. Fairhurst, and D. Bisset. Investigating feedforward neural networks with respect to the rejection of spurious patterns. *Pattern Recognition Letters*, 16(2):207–212, feb 1995. ISSN 01678655. doi: 10.1016/0167-8655(94)00092-H.

[141] Sambu Seo, M. Wallat, T. Graepel, and K. Obermayer. Gaussian process regression: active data selection and test point rejection. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pages 241–246. IEEE, 2000. ISBN 0-7695-0619-4. doi: 10.1109/IJCNN.2000.861310.

[142] H. Hosseini, Y. Chen, S. Kannan, B. Zhang, and R. Poovendran. Blocking Transferability of Adversarial Examples in Black-Box Learning Systems. *arXiv preprint arXiv:1703.04318*, 2017.

[143] X. Wang and S. M. Yiu. Classification with Rejection: Scaling Generative Classifiers with Supervised Deep Infomax. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 2980–2986, California, jul 2020. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-6-5. doi: 10.24963/ijcai.2020/412.

[144] A. Sotgiu, A. Demontis, M. Melis, B. Biggio, G. Fumera, X. Feng, and F. Roli. Deep neural rejection against adversarial examples. *EURASIP Journal on Information Security*, 2020(1):5, dec 2020. ISSN 2510-523X. doi: 10.1186/s13635-020-00105-y.

[145] K. Urahama and Y. Furukawa. Gradient descent learning of nearest neighbor classifiers with outlier rejection. *Pattern Recognition*, 28(5): 761–768, may 1995. ISSN 00313203. doi: 10.1016/0031-3203(94)00142-9.

[146] F. Lotte, H. Mouchère, and A. Lécuyer. Pattern rejection strategies for the design of self-paced EEG-based brain-computer interfaces. *Proceedings - International Conference on Pattern Recognition*, pages 6–10, 2008. ISSN 10514651. doi: 10.1109/icpr.2008.4761454.

[147] Y. C. Hsu, Y. Shen, H. Jin, and Z. Kira. Generalized ODIN: Detecting Out-of-Distribution Image without Learning from Out-of-Distribution Data. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 10948–10957, 2020. ISSN 10636919. doi: 10.1109/CVPR42600.2020.01096.

[148] T. C. Landgrebe, D. M. Tax, P. Paclík, R. P. Duin, and C. Andrew. A combining strategy for ill-defined problems. In *Proc. of the 15th Annual Symposium of the Pattern Recognition Association of South Africa*, pages 57–62, 2004.

[149] W. Homenda, M. Luckner, and W. Pedrycz. Classification with rejection based on various SVM techniques. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 3480–3487. IEEE, jul 2014. ISBN 978-1-4799-1484-5. doi: 10.1109/IJCNN.2014.6889655.

[150] L. K. Hansen, C. Liisberg, and P. Salamon. The Error-Reject Tradeoff. *Open Systems & Information Dynamics*, 4(2):159–184, 1997. ISSN 1573-1324. doi: 10.1023/A:1009643503022.

[151] R. El-Yaniv and Y. Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11:1605–1641, 2010. ISSN 15324435.

[152] 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*, pages 93–104, 2000. ISBN 1581132182. doi: 10.1016/s0020-7292(09)60373-8.

[153] D. M. Tax and R. P. Duin. Growing a multi-class classifier with a reject option. *Pattern Recognition Letters*, 29(10):1565–1570, jul 2008. ISSN 01678655. doi: 10.1016/j.patrec.2008.03.010.

[154] C. Sutton and A. Mccallum. An Introduction to Conditional Random Fields for Relational Learning. *Introduction to Statistical Relational Learning*, 2019. doi: 10.7551/mitpress/7432.003.0006.

[155] G. Harshvardhan, M. K. Gourisaria, M. Pandey, and S. S. Rautaray. A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review*, 38:100285, 2020. ISSN 15740137. doi: 10.1016/j.cosrev.2020.100285.

[156] C. Sutton and A. McCallum. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 4(4):267–373, 2011. ISSN 19358237. doi: 10.1561/2200000013.

[157] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. *ACM SIGMOD Record*, 29:427–438, 2000. doi: 10.1145/342009.335437.

[158] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, volume 96, pages 226–231, 1996. ISBN 9780444527011. doi: 10.1016/B978-044452701-1.00067-3.

[159] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Piatt. Support vector method for novelty detection. *Advances in Neural Information Processing Systems*, 12:582–588, 1999. ISSN 10495258.

[160] V. Cohen-Addad, S. Lattanzi, A. Norouzi-Fard, C. Sohler, and O. Svensson. Fast and Accurate k-means++ via Rejection Sampling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[161] C. De Stefano, C. Sansone, and M. Vento. To reject or not to reject: that is the question-an answer in case of neural classifiers. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 30 (1):84–94, 2000. ISSN 10946977. doi: 10.1109/5326.827457.

[162] W. Homenda, M. Luckner, and W. Pedrycz. Classification with rejection: Concepts and evaluations. *Advances in Intelligent Systems and Computing*, 364(December):413–425, 2016. ISSN 21945357. doi: 10.1007/978-3-319-19090-7_31.

[163] F. Condessa, J. Bioucas-Dias, and J. Kovačević. Performance measures for classification systems with rejection. *Pattern Recognition*, 63:437–450, mar 2017. ISSN 00313203. doi: 10.1016/j.patcog.2016.10.011.

[164] H. Jiang, B. Kim, M. Y. Guan, and M. Gupta. To Trust Or Not To Trust A Classifier. In *32nd Conference on Neural Information Processing Systems (NIPS)*, pages 5541–5552, Montreal, Canada, may 2018.

[165] D. Madras, T. Pitassi, and R. Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer. *Advances in Neural Information Processing Systems*, 2018-Decem(NeurIPS):6147–6157, nov 2018. ISSN 10495258.

[166] D. P. Mesquita, L. S. Rocha, J. P. P. Gomes, and A. R. da Rocha Neto. Classification with reject option for software defect prediction. *Applied Soft Computing Journal*, 49:1085–1093, 2016. ISSN 15684946. doi: 10.1016/j.asoc.2016.06.023.

[167] I. Pillai, G. Fumera, and F. Roli. A Classification Approach with a Reject Option for Multi-label Problems. In *Lecture Notes in Computer Science*, volume 6978 LNCS, pages 98–107. Springer Berlin Heidelberg, 2011. ISBN 9783642240843. doi: 10.1007/978-3-642-24085-0_11.

[168] I. Pillai, G. Fumera, and F. Roli. Multi-label classification with a reject option. *Pattern Recognition*, 46(8):2256–2266, aug 2013. ISSN 00313203. doi: 10.1016/j.patcog.2013.01.035.

[169] H. Zhang, A. Li, J. Guo, and Y. Guo. Hybrid models for open set recognition. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 102–117, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58580-8.

[170] K. Lee, K. Lee, H. Lee, and J. Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[171] D. Lin, L. Sun, K. A. Toh, J. B. Zhang, and Z. Lin. Twin SVM with a reject option through ROC curve. *Journal of the Franklin Institute*, 355 (4):1710–1732, 2018. ISSN 00160032. doi: 10.1016/j.jfranklin.2017.05.003.

[172] M. S. A. Nadeem, J. Zucker, and B. Hanczar. Accuracy-Rejection Curves (ARCs) for Comparing Classification Methods with a Reject Option. *Machine Learning in Systems Biology*, 8:65–81, 2010.

[173] A. R. da Rocha Neto, R. Sousa, G. d. A. Barreto, and J. S. Cardoso. Diagnostic of Pathology on the Vertebral Column with Embedded Reject Option. In *IbPRIA 2011: Pattern Recognition and Image Analysis*, volume 6669 LNCS, pages 588–595, 2011. ISBN 9783642212567. doi: 10.1007/978-3-642-21257-4_73.

[174] Z. Zidelmal, A. Amirou, and A. Belouchrani. Heartbeat classifcation using support vector machines (SVMs) with an embedded reject option. *International Journal of Pattern Recognition and Artificial Intelligence*, 26 (01):1250001, feb 2012. ISSN 0218-0014. doi: 10.1142/S0218001412500012.

[175] R. Gamelas Sousa and J. S. Cardoso. The data replication method for the classification with reject option. *AI Communications*, 26(3):281–302, nov 2013. ISSN 09217126. doi: 10.3233/AIC-130566.

[176] L. Fischer, D. Nebel, T. Villmann, B. Hammer, and H. Wersing. Rejection Strategies for Learning Vector Quantization – A Comparison of Probabilistic and Deterministic Approaches. In *22nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2014 - Proceedings*, pages 109–118. Springer Cham, 2014. ISBN 9782874190957. doi: 10.1007/978-3-319-07695-9_10.

[177] L. Fischer, B. Hammer, and H. Wersing. Local Rejection Strategies for Learning Vector Quantization. In *Lecture Notes in Computer Science*, volume 8681 LNCS, pages 563–570. Springer Cham, 2014. ISBN 9782874190957. doi: 10.1007/978-3-319-11179-7_71.

[178] B. Hanczar and M. Sebag. Combination of one-class support vector machines for classification with reject option. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8724 LNAI(PART 1):547–562, 2014. ISSN 16113349. doi: 10.1007/978-3-662-44848-9_35.

[179] J. Brinkrolf and B. Hammer. Probabilistic extension and reject options for pairwise LVQ. In *2017 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM)*, pages 1–8. IEEE, jun 2017. ISBN 978-1-5090-6638-4. doi: 10.1109/WSOM.2017.8020028.

[180] J. P. Göpfert, B. Hammer, and H. Wersing. Mitigating concept drift via rejection. In *Lecture Notes in Computer Science*, volume 11139

LNCS, pages 456–467. Springer Cham, 2018. ISBN 9783030014179. doi: 10.1007/978-3-030-01418-6_45.

[181] J. Brinkrolf and B. Hammer. Interpretable machine learning with reject option. *Automatisierungstechnik*, 66(4):283–290, 2018. doi: 10.1515/auto-2017-0123.

[182] S. Shekhar, M. Ghavamzadeh, and T. Javidi. Binary Classification with Bounded Abstention Rate. *arXiv preprint arXiv:1905.09561*, pages 1–35, may 2019.

[183] C. Ni, N. Charoenphakdee, J. Honda, and M. Sugiyama. On the Calibration of Multiclass Classification with Rejection. *arXiv preprint arXiv:1901.10655*, pages 1–31, jan 2019.

[184] H. Mozannar and D. Sontag. Consistent Estimators for Learning to Defer to an Expert. *arXiv preprint arXiv:2006.01862*, 2020.

[185] C. Ferri and J. Hernández-Orallo. Cautious Classifiers. *Proceedings of ROC Analysis in Artificial Intelligence, 1st International Workshop (ROCAI-2004)*, pages 27–36, 2004.

[186] G. Fumera, I. Pillai, and F. Roli. A two-stage classifier with reject option for text categorisation. In *Lecture Notes in Computer Science*, volume 3138, pages 771–779. Springer Berlin Heidelberg, 2004. ISBN 9783540225706. doi: 10.1007/978-3-540-27868-9_84.

[187] V.-L. Nguyen and E. Hullermeier. Reliable Multilabel Classification: Prediction with Partial Abstention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5264–5271, apr 2020. doi: 10.1609/aaai.v34i04.5972.

[188] M. Golfarelli, D. Maio, and D. Malton. On the error-reject trade-off in biometric verification systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):786–796, jul 1997. ISSN 01628828. doi: 10.1109/34.598237.

[189] L. Cordella, C. De Stefano, F. Fontanella, and A. S. Scotto Di Freca. Random forest for reliable pre-classification of handwritten characters. In *Proceedings - International Conference on Pattern Recognition*, pages 1319–1324. IEEE, 2014. ISBN 9781479952083. doi: 10.1109/ICPR.2014.236.

[190] J. Heo, H. B. Lee, S. Kim, J. Lee, K. J. Kim, E. Yang, and S. J. Hwang. Uncertainty-Aware Attention for Reliable Interpretation and Prediction. In *Advances in Neural Information Processing Systems*, volume 31, pages 909–918, Montreal, Canada, 2018. Curran Associates, Inc.

[191] Y. Geifman and R. El-Yaniv. SelectiveNet: A Deep Neural Network with an Integrated Reject Option. In *International Conference on Machine Learning (ICML 2019)*, volume 2019-June, pages 3768–3776, jan 2019. ISBN 9781510886988.

[192] C. Denis, M. Hebiri, and A. Zaoui. Regression with reject option and application to kNN. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20073–20082. Curran Associates, Inc., jun 2020.

[193] A. Asif and F. u. A. A. Minhas. Generalized Neural Framework for Learning with Rejection. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, jul 2020. ISBN 978-1-7281-6926-2. doi: 10.1109/IJCNN48605.2020.9206612.

[194] Q. Wu, C. Jia, and W. Chen. A Novel Classification-Rejection Sphere SVMs for Multi-class Classification Problems. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 1, pages 34–38. IEEE, 2007. ISBN 0-7695-2875-9. doi: 10.1109/ICNC.2007.109.

[195] L. Cordella, C. De Stefano, F. Tortorella, and M. Vento. A method for improving classification reliability of multilayer perceptrons. *IEEE Transactions on Neural Networks*, 6(5):1140–1147, 1995. ISSN 10459227. doi: 10.1109/72.410358.

[196] F. Tortorella. An Optimal Reject Rule for Binary Classifiers. In *Lecture Notes in Computer Science*, volume 1876 LNCS, pages 611–620. Springer Cham, 2000. ISBN 3540679464. doi: 10.1007/3-540-44522-6_63.

[197] A. Bounsiar, P. Beauseroy, and E. Grall-Maës. General solution and learning method for binary classification with performance constraints. *Pattern Recognition Letters*, 29(10):1455–1465, 2008. ISSN 01678655. doi: 10.1016/j.patrec.2008.02.025.

[198] C. C. Friedel, U. Rückert, and S. Kramer. Cost Curves for Abstaining Classifiers. In *ICML 2006 - 3th workshop ROC Analysis in ML*, Pittsburgh, Pennsylvania, 2006.

[199] B. Hanczar. Performance visualization spaces for classification with rejection option. *Pattern Recognition*, 96:106984, dec 2019. ISSN 00313203. doi: 10.1016/j.patcog.2019.106984.

[200] 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.

IEEE Access, 7:160125–160143, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2950244.

[201] L. Coenen, A. K. Abdullah, and T. Guns. Probability of default estimation, with a reject option. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 439–448. IEEE, oct 2020. ISBN 978-1-7281-8206-3. doi: 10.1109/DSAA49011.2020.00058.

[202] T. C. Landgrebe, D. M. Tax, P. Paclík, and R. P. Duin. The interaction between classification and reject performance for distance-based reject-option classifiers. *Pattern Recognition Letters*, 27(8):908–917, jun 2006. ISSN 01678655. doi: 10.1016/j.patrec.2005.10.015.

[203] D. Van der Plas, W. Meert, J. Verbraecken, and J. Davis. A reject option for automated sleep stage scoring. In *ICML Workshop on Interpretable Machine Learning in Healthcare*, Virtual event, 2021.

[204] R. Herbei and M. H. Wegkamp. Classification with reject option. *Canadian Journal of Statistics*, 34(4):709–721, dec 2006. ISSN 03195724. doi: 10.1002/cjs.5550340410.

[205] Y. Grandvalet, A. Rakotomamonjy, J. Keshet, and S. Canu. Support Vector Machines with a Reject Option. In *Advances in Neural Information Processing Systems 21*, pages 537–544. Curran Associates, Inc., 2009.

[206] T. M. Ha. An optimum class-selective rejection rule for pattern recognition. In *Proceedings of 13th International Conference on Pattern Recognition*, volume 2, pages 75–80 vol.2. IEEE, 1996. ISBN 0-8186-7282-X. doi: 10.1109/ICPR.1996.546727.

[207] R. Battiti and A. M. Colla. Democracy in neural nets: Voting schemes for classification. *Neural Networks*, 7(4):691–707, jan 1994. ISSN 08936080. doi: 10.1016/0893-6080(94)90046-9.

[208] N. Hatami and C. Chira. Classifiers with a reject option for early time-series classification. *Proceedings of the 2013 IEEE Symposium on Computational Intelligence and Ensemble Learning, CIEL 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI 2013*, pages 9–16, 2013. doi: 10.1109/CIEL.2013.6613134.

[209] R. Gamelas Sousa, A. da Rocha Neto, J. S. R.Cardoso, and G. A. Barreto. Robust classification with reject option using the self-organizing map. *Neural Computing and Applications*, 26(7):1603–1619, oct 2015. ISSN 0941-0643. doi: 10.1007/s00521-015-1822-2.

[210] P. Foggia, C. Sansone, F. Tortorella, and M. Vento. Multiclassification: reject criteria for the Bayesian combiner. *Pattern Recognition*, 32(8):1435–1447, aug 1999. ISSN 00313203. doi: 10.1016/S0031-3203(98)00169-1.

[211] L. Lam and C. Y. Suen. Optimal combinations of pattern classifiers. *Pattern Recognition Letters*, 16(9):945–954, sep 1995. ISSN 01678655. doi: 10.1016/0167-8655(95)00050-Q.

[212] B. Krawczyk and A. Cano. Online ensemble learning with abstaining classifiers for drifting and noisy data streams. *Applied Soft Computing*, 68:677–692, jul 2018. ISSN 15684946. doi: 10.1016/j.asoc.2017.12.008.

[213] M. A. Kocak, D. Ramirez, E. Erkip, and D. Shasha. SafePredict: A Meta-Algorithm for Machine Learning That Uses Refusals to Guarantee Correctness. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(2):1–1, 2020. ISSN 0162-8828. doi: 10.1109/TPAMI.2019.2932415.

[214] C. Sansone, F. Tortorella, and M. Vento. A classification reliability driven reject rule for multi-expert systems. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(06):885–904, sep 2001. ISSN 0218-0014. doi: 10.1142/S0218001401001210.

[215] C. Dubos, S. Bernard, S. Adam, and R. Sabourin. ROC-based cost-sensitive classification with a reject option. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3320–3325. IEEE, dec 2016. ISBN 978-1-5090-4847-2. doi: 10.1109/ICPR.2016.7900146.

[216] P. Simeone, C. Marrocco, and F. Tortorella. Design of reject rules for ECOC classification systems. *Pattern Recognition*, 45(2):863–875, aug 2012. ISSN 00313203. doi: 10.1016/j.patcog.2011.08.001.

[217] Y. Le Cun, O. Matan, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jacket, and H. Baird. Handwritten zip code recognition with multilayer networks. In *[1990] Proceedings. 10th International Conference on Pattern Recognition*, volume ii, pages 35–40. IEEE Comput. Soc. Press, 1990. ISBN 0-8186-2062-5. doi: 10.1109/ICPR.1990.119325.

[218] A. Lewicke, E. S. Sazonov, M. Corwin, M. Neuman, and S. Schuckers. Sleep Versus Wake Classification From Heart Rate Variability Using Computational Intelligence: Consideration of Rejection in Classification Models. *IEEE Transactions on Biomedical Engineering*, 55(1):108–118, jan 2008. ISSN 0018-9294. doi: 10.1109/TBME.2007.900558.

[219] Y. Geifman and R. El-Yaniv. Selective Classification for Deep Neural Networks. In *Advances in Neural Information Processing Systems*, pages 4879–4888. Curran Associates, Inc., may 2017.

[220] K. R. Varshney. A risk bound for ensemble classification with a reject option. In *2011 IEEE Statistical Signal Processing Workshop (SSP)*, pages 769–772. IEEE, jun 2011. ISBN 978-1-4577-0569-4. doi: 10.1109/SSP.2011.5967817.

[221] Z. Lou, K. Liu, J. Yang, and C. Suen. Rejection Criteria and Pairwise Discrimination of Handwritten Numerals Based on Structural Features. *Pattern Analysis & Applications*, 2(3):228–238, aug 1999. ISSN 1433-7541. doi: 10.1007/s100440050031.

[222] J. Suutala, S. Pirttikangas, J. Riekki, and J. Röning. Reject-Optional LVQ-Based Two-Level Classifier to Improve Reliability in Footstep Identification. In *Lecture Notes in Computer Science*, volume 3001, pages 182–187. Springer Berlin Heidelberg, 2004. ISBN 9783540218357. doi: 10.1007/978-3-540-24646-6_12.

[223] L. Fischer, B. Hammer, and H. Wersing. Efficient rejection strategies for prototype-based classification. *Neurocomputing*, 169:334–342, dec 2015. ISSN 09252312. doi: 10.1016/j.neucom.2014.10.092.

[224] A. Ahmadi, S. Omatu, T. Fujinaka, and T. Kosaka. Improvement of reliability in banknote classification using reject option and local PCA. *Information Sciences*, 168(1-4):277–293, dec 2004. ISSN 00200255. doi: 10.1016/j.ins.2004.02.018.

[225] J. Mena, O. Pujol, and J. Vitria. Uncertainty-Based Rejection Wrappers for Black-Box Classifiers. *IEEE Access*, 8:101721–101746, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.2996495.

[226] K. Joana, M. Christian, and G. Clemens. Securing Deep Learning Models with Autoencoder based Anomaly Detection. In *Proceedings of the European Conference of the PHM Society 2021*, pages 221–233, Virtual event, 2021. PHM Society. ISBN 9781936263349. doi: 10.36001/phme.2021.v6i1.2841.

[227] H. Mizutani. Discriminative learning for minimum error and minimum reject classification. In *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170)*, volume 1, pages 136–140, Brisbane, Queensland, Australia, Australia, 1998. IEEE Comput. Soc. ISBN 0-8186-8512-3. doi: 10.1109/ICPR.1998.711099.

[228] C. Ma, M. A. Randolph, and J. Drish. A support vector machines-based rejection technique for speech recognition. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 1: 381–384, 2001. ISSN 15206149. doi: 10.1109/icassp.2001.940847.

[229] M. J. Pazzani, P. Murphy, K. Ali, and D. Schulenburg. Tradingoff coverage for accuracy in forecasts: Applications to clinical data analysis. In *Proceedings of the AAAI Symposium on AI in Medicine*, pages 100–104, 1994.

[230] F. Condessa, J. Bioucas-Dias, C. A. Castro, J. Ozolek, and J. Kovacevic. Classification with reject option using contextual information. In *2013 IEEE 10th International Symposium on Biomedical Imaging*, pages 1340–1343, San Francisco, California, apr 2013. IEEE. ISBN 978-1-4673-6455-3. doi: 10.1109/ISBI.2013.6556780.

[231] F. Condessa, J. Bioucas-Dias, C. A. Castro, J. Ozolek, and J. Kovačević. Image Classification with Rejection using Contextual Information. *arXiv preprint arXiv:1509.01287*, pages 1–21, 2015.

[232] F. Condessa, J. Bioucas-Dias, and J. Kovacevic. Supervised hyperspectral image classification with rejection. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(6):2321–2332, jun 2015. ISSN 1939-1404. doi: 10.1109/IGARSS.2015.7326344.

[233] F. Condessa, J. Bioucas-Dias, and J. Kovacevic. Robust hyperspectral image classification with rejection fields. In *Workshop on Hyperspectral Image and Signal Processing, Evolution in Remote Sensing*, volume 2015-June, pages 1–4, Tokyo, Japan, jun 2015. IEEE. ISBN 9781467390156. doi: 10.1109/WHISPERS.2015.8075465.

[234] J. Lei. Classification with confidence. *Biometrika*, 101(4):755–769, 2014. ISSN 14643510. doi: 10.1093/biomet/asu038.

[235] S. Laroui, X. Descombes, A. Vernay, F. Villiers, E. Debreuve, S. Laroui, X. Descombes, A. Vernay, F. Villiers, F. Villalba, and S. Laroui. How to define a rejection class based on model learning? In *25th International Conference on Pattern Recogniton*, Milano, 2021.

[236] E. Gasca A., S. Saldaña T., J. S. Sánchez G., V. Velásquez G., E. Rendón L., I. M. Abundez B., R. M. Valdovinos R., and R. Cruz R. A Rejection Option for the Multilayer Perceptron Using Hyperplanes. In *Lecture Notes in Computer Science*, volume 6593 LNCS, pages 51–60. Springer Berlin Heidelberg, 2011. ISBN 9783642202810. doi: 10.1007/978-3-642-20282-7_6.

[237] S. Berlemont, G. Lefebvre, S. Duffner, and C. Garcia. Siamese neural network based similarity metric for inertial gesture classification and rejection. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, pages 1–6. IEEE, may 2015. ISBN 978-1-4799-6026-2. doi: 10.1109/FG.2015.7163112.

[238] T. M. Ha. The optimum class-selective rejection rule. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):608–615, jun 1997. ISSN 01628828. doi: 10.1109/34.601248.

[239] T. M. Ha. Optimum tradeoff between class-selective rejection error and average number of classes. *Engineering Applications of Artificial Intelligence*, 10(6):525–529, dec 1997. ISSN 09521976. doi: 10.1016/S0952-1976(97)00034-1.

[240] H. Le Capitaine and C. Frelicot. An Optimum Class-Rejective Decision Rule and Its Evaluation. In *2010 20th International Conference on Pattern Recognition*, pages 3312–3315. IEEE, aug 2010. ISBN 978-1-4244-7542-1. doi: 10.1109/ICPR.2010.810.

[241] G. Fumera, I. Pillai, and F. Roli. Classification with reject option in text categorisation systems. In *12th International Conference on Image Analysis and Processing, 2003.Proceedings.*, pages 582–587. IEEE Comput. Soc, 2003. ISBN 0-7695-1948-2. doi: 10.1109/ICIAP.2003.1234113.

[242] H. Le Capitaine and C. Frélicot. A family of measures for best top-n class-selective decision rules. *Pattern Recognition*, 45(1):552–562, 2012. ISSN 00313203. doi: 10.1016/j.patcog.2011.04.030.

[243] F. Tortorella. Reducing the classification cost of support vector classifiers through an ROC-based reject rule. *Pattern Analysis and Applications*, 7 (2):128–143, jul 2004. ISSN 1433-7541. doi: 10.1007/s10044-004-0209-2.

[244] R. Zhang and D. Metaxas. RO-SVM: Support Vector Machine with Reject Option for Image Categorization. In *Procedings of the British Machine Vision Conference 2006*, pages 123.1–123.10. British Machine Vision Association, 2006. ISBN 1-901725-32-4. doi: 10.5244/C.20.123.

[245] K. R. Varshney. A Kernel Based Rejection Method for Supervised Classification. *International Journal of Computational Intelligence*, 3 (4):312–321, 2006.

[246] A. Balsubramani. Learning to Abstain from Binary Prediction. *arXiv preprint arXiv:1602.08151*, pages 1–23, 2016.

[247] K. Hamid, A. Asif, W. Abbasi, D. Sabih, and F. A. Afsar. Machine Learning with Abstention for Automated Liver Disease Diagnosis. In *2017 International Conference on Frontiers of Information Technology (FIT)*, volume 2017-Janua, pages 356–361. IEEE, dec 2017. ISBN 978-1-5386-3567-4. doi: 10.1109/FIT.2017.00070.

[248] C. Cortes, G. DeSalvo, C. Gentile, M. Mohri, and S. Yang. Online Learning with Abstention. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1059–1067, Stockholmsmässan, Stockholm Sweden, 2018. PMLR. ISBN 9781510867963.

[249] J. T. Kwok. Moderating the outputs of support vector machine classifiers. *IEEE Transactions on Neural Networks*, 10(5):1018–1031, 1999. doi: 10.1109/IJCNN.1999.831080.

[250] W. Tang and E. S. Sazonov. Highly Accurate Recognition of Human Postures and Activities Through Classification With Rejection. *IEEE Journal of Biomedical and Health Informatics*, 18(1):309–315, jan 2014. ISSN 2168-2194. doi: 10.1109/JBHI.2013.2287400.

[251] R. Gamelas Sousa, A. R. da Rocha Neto, G. A. Barreto, J. S. Cardoso, and M. T. Coimbra. Reject option paradigm for the reduction of support vectors. *22nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2014 - Proceedings*, pages 331–336, 2014.

[252] D. Conte, P. Foggia, G. Percannella, A. Saggese, and M. Vento. An ensemble of rejecting classifiers for anomaly detection of audio events. In *Proceedings - 2012 IEEE 9th International Conference on Advanced Video and Signal-Based Surveillance, AVSS 2012*, pages 76–81. IEEE, 2012. ISBN 9780769547978. doi: 10.1109/AVSS.2012.9.

[253] T. Villmann, M. Kaden, D. Nebel, and M. Biehl. Learning Vector Quantization with Adaptive Cost-Based Outlier-Rejection. In *Lecture Notes in Computer Science*, volume 9257, pages 772–782. Springer, Cham, 2015. ISBN 9783319231167. doi: 10.1007/978-3-319-23117-4_66.

[254] J. Arlandis, J. C. Perez-Cortes, and J. Cano. Rejection strategies and confidence measures for a k-NN classifier in an OCR task. In *International Conference on Pattern Recognition*, volume 16, pages 576–579, 2002. doi: 10.1109/icpr.2002.1044806.

[255] E. Merényi, M. J. Mendenhall, and P. O'Driscoll. Self-Adjusting Reject Options in Prototype Based Classification. In *Advances in Intelligent Systems and Computing*, volume 428, 2016. ISBN 9783319285177. doi: 10.1007/978-3-319-28518-4.

[256] C. Zhang, W. Wang, and X. Qiao. On Reject and Refine Options in Multicategory Classification. *Journal of the American Statistical Association*, 113(522):730–745, apr 2018. ISSN 0162-1459. doi: 10.1080/01621459.2017.1282372.

[257] R. Gamelas Sousa, A. R. da Rocha Neto, J. S. Cardoso, and G. A. Barreto. Classification with Reject Option Using the Self-Organizing Map. In *Lecture Notes in Computer Science*, volume 8681 LNCS, chapter Artificial, pages 105–112. Springer, Cham, 2014. ISBN 9783319111780. doi: 10.1007/978-3-319-11179-7_14.

[258] R. El-Yaniv and Y. Wiener. Agnostic selective classification. *Advances in Neural Information Processing Systems*, 24:1–9, 2011.

[259] C. Marrocco, M. Molinara, and F. Tortorella. An empirical comparison of ideal and empirical ROC-based reject rules. In *Lecture Notes in Computer Science*, volume 4571 LNAI, pages 47–60. Springer, Berlin, Heidelberg, 2007. ISBN 9783540734987. doi: 10.1007/978-3-540-73499-4_5.

[260] R. Gamelas Sousa, B. Mora, and J. S. Cardoso. An ordinal data method for the classification with reject option. *8th International Conference on Machine Learning and Applications, ICMLA 2009*, 2:746–750, 2009. doi: 10.1109/ICMLA.2009.11.

[261] F. Tortorella. A ROC-based reject rule for dichotomizers. *Pattern Recognition Letters*, 26(2):167–180, 2005. ISSN 01678655. doi: 10.1016/j.patrec.2004.09.004.

[262] H. Cecotti and S. Vajda. Rejection Schemes in Multi-class Classification – Application to Handwritten Character Recognition. In *2013 12th International Conference on Document Analysis and Recognition*, pages 445–449. IEEE, aug 2013. ISBN 978-0-7695-4999-6. doi: 10.1109/ICDAR.2013.96.

[263] C. M. Santos-Pereira and A. M. Pires. On optimal reject rules and ROC curves. *Pattern Recognition Letters*, 26(7):943–952, 2005. ISSN 01678655. doi: 10.1016/j.patrec.2004.09.042.

[264] T. Pietraszek. Optimizing abstaining classifiers using ROC analysis. In *Proceedings of the 22nd international conference on Machine learning - ICML '05*, pages 665–672, New York, New York, USA, 2005. ACM Press. ISBN 1595931805. doi: 10.1145/1102351.1102435.

[265] T. Pietraszek. On the use of ROC analysis for the optimization of abstaining classifiers. *Machine Learning*, 68(2):137–169, jul 2007. ISSN 0885-6125. doi: 10.1007/s10994-007-5013-y.

[266] X. Jigang, Q. Zhengding, and W. Jie. Bootstrap methods for reject rules of fisher LDA. *Proceedings - International Conference on Pattern Recognition*, 3:425–428, 2006. ISSN 10514651. doi: 10.1109/ICPR.2006.338.

[267] E.-h. Zheng, C. Zou, J. Sun, and L. Chen. Cost-sensitive SVM with Error Cost and Class-dependent Reject Cost. *International Journal of Computer Theory and Engineering*, 3(1):130–135, 2011. ISSN 17938201. doi: 10.7763/IJCTE.2011.V3.294.

[268] G. Fumera, F. Roli, and G. Giacinto. Multiple reject thresholds for improving classification reliability. In *Lecture Notes in Computer Science*, volume 1876 LNCS, pages 863–871. Springer Berlin Heidelberg, 2000. ISBN 3540679464. doi: 10.1007/3-540-44522-6_89.

[269] R. Muzzolini, Y.-H. Yang, and R. Pierson. Classifier design with incomplete knowledge. *Pattern Recognition*, 31(4):345–369, apr 1998. ISSN 00313203. doi: 10.1016/S0031-3203(97)00056-3.

[270] L. Fischer, B. Hammer, and H. Wersing. Optimum Reject Options for Prototype-based Classification. *arXiv preprint arXiv:1503.06549*, mar 2015.

[271] L. Fischer, B. Hammer, and H. Wersing. Optimal local rejection for classifiers. *Neurocomputing*, 214:445–457, nov 2016. ISSN 09252312. doi: 10.1016/j.neucom.2016.06.038.

[272] M. Perello-Nieto, T. M. Filho, M. Kull, and P. Flach. Background check: A general technique to build more reliable and versatile classifiers. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 1143–1148, 2017. ISBN 9781509054725. doi: 10.1109/ICDM.2016.125.

[273] G. Fumera and F. Roli. Support Vector Machines with Embedded Reject Option. In *Pattern Recognition with Support Vector Machines*, pages 68–82, Berlin, Heidelberg, feb 2002. Springer Berlin Heidelberg. doi: 10.1007/3-540-45665-1_6.

[274] J. S. Cardoso and J. F. Pinto Da Costa. Learning to classify ordinal data: The data replication method. *Journal of Machine Learning Research*, 8: 1393–1429, 2007. ISSN 15324435.

[275] G. Vasconcelos, M. Fairhurst, and D. Bisset. Enhanced reliability of multilayer perceptron networks through controlled pattern rejection. *Electronics Letters*, 29(3):261, 1993. ISSN 00135194. doi: 10.1049/el:19930179.

[276] P. L. Bartlett and M. H. Wegkamp. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9:1823–1840, 2008. ISSN 15324435. doi: 10.5555/1390681.1442792.

[277] M. Yuan and M. H. Wegkamp. Classification methods with reject option based on convex risk minimization. *Journal of Machine Learning Research*, 11:111–130, 2010. ISSN 15324435.

[278] N. Charoenphakdee, Z. Cui, Y. Zhang, and M. Sugiyama. Classification with Rejection Based on Cost-sensitive Classification. In *Proceedings of the 38th International Conference on Machine Learning*, pages 1507–1517, 2021.

[279] P. X. Loeffel, C. Marsala, and M. Detyniecki. Classification with a reject option under Concept Drift: The Droplets algorithm. In *Proceedings of the 2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015*, pages 1–9, 2015. ISBN 9781467382731. doi: 10.1109/DSAA.2015.7344808.

[280] S. Singh and M. Markou. An Approach to Novelty Detection Applied to the Classification of Image Regions. *IEEE Transactions on Knowledge and Data Engineering*, 16(4):396–407, 2004. ISSN 10414347. doi: 10.1109/TKDE.2004.1269665.

[281] C. Frélicot. Learning rejection thresholds for a class of fuzzy classifiers from possibilistic clustered noisy data. In *7th International Fuzzy Systems Association World Congress, IFSA*, volume 3, pages 111–116, 1997.

[282] C. Frélicot. On unifying probabilistic/fuzzy and possibilistic rejection-based classifiers. In *SSPR /SPR 1998: Advances in Pattern Recognition*, pages 736–745. Springer Berlin Heidelberg, 1998. doi: 10.1007/BFb0033298.

[283] C. Frélicot and L. Mascarilla. Reject strategies driven combination of pattern classifiers. *Pattern Analysis and Applications*, 5(2):234–243, 2002. ISSN 14337541. doi: 10.1007/s100440200021.

[284] T. Horiuchi. Class-selective rejection rule to minimize the maximum distance between selected classes. *Pattern Recognition*, 31(10):1579–1588, oct 1998. ISSN 00313203. doi: 10.1016/S0031-3203(97)00136-2.

[285] B. Hanczar and E. R. Dougherty. Classification with reject option in gene expression data. *Bioinformatics*, 24(17):1889–1895, sep 2008. ISSN 1367-4803. doi: 10.1093/bioinformatics/btn349.

[286] M. Li and I. K. Sethi. Confidence-based classifier design. *Pattern Recognition*, 39(7):1230–1240, jul 2006. ISSN 00313203. doi: 10.1016/j.patcog.2006.01.010.

[287] T. Pang, H. Zhang, D. He, Y. Dong, H. Su, W. Chen, J. Zhu, and T.-Y. Liu. Adversarial Training with Rectified Rejection. *arXiv preprint arXiv:2105.14785*, pages 1–23, 2021.

[288] A. Sayedi, M. Zadimoghaddam, and A. Blum. Trading off mistakes and don't-know predictions. *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010, NIPS 2010*, pages 1–9, 2010.

[289] D. Boulegane, A. Bifet, and G. Madhusudan. Arbitrated Dynamic Ensemble with Abstaining for Time-Series Forecasting on Data Streams. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1040–1045. IEEE, dec 2019. ISBN 978-1-7281-0858-2. doi: 10.1109/BigData47090.2019.9005541.

[290] C. Cortes, G. DeSalvo, and M. Mohri. Learning with Rejection. In *Proceedings of The 27th International Conference on Algorithmic Learning Theory (ALT 2016)*, volume 9925 LNAI, pages 67–82, Bari, Italy, 2016. Springer Berlin Heidelberg. ISBN 9783319463780. doi: 10.1007/978-3-319-46379-7_5.

[291] M. H. Wegkamp. Lasso type classifiers with a reject option. *Electronic Journal of Statistics*, 1:155–168, may 2007. ISSN 1935-7524. doi: 10.1214/07-EJS058.

[292] M. H. Wegkamp and M. Yuan. Support vector machines with a reject option. *Bernoulli*, 17(4):1368–1385, jan 2012. ISSN 13507265. doi: 10.3150/10-BEJ320.

[293] C. Cortes, G. DeSalvo, and M. Mohri. Boosting with abstention. In *Advances in Neural Information Processing Systems*, pages 1668–1676, 2016.

[294] S. Thulasidasan, T. Bhattacharya, J. Bilmes, G. Chennupati, and J. Mohd-Yusof. Combating Label Noise in Deep Learning Using Abstention. In *Proceedings of the 36 th International Conference on Machine Learning*, 2019.

[295] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, dec 2008. ISBN 978-0-7695-3502-9. doi: 10.1109/ICDM.2008.17.

[296] H. Guan, Y. Zhang, H. D. Cheng, and X. Tang. Bounded-abstaining classification for breast tumors in imbalanced ultrasound images. *International Journal of Applied Mathematics and Computer Science*, 30(2):325–336, 2020. ISSN 20838492. doi: 10.34768/amcs-2020-0025.

[297] S. Kang, S. Cho, S.-j. Rhee, and K.-S. Yu. Reliable prediction of anti-diabetic drug failure using a reject option. *Pattern Analysis and Applications*, 20(3):883–891, aug 2017. ISSN 1433-7541. doi: 10.1007/s10044-016-0585-4.

[298] G. Fumera and F. Roli. Analysis of error-reject trade-off in linearly combined multiple classifiers. *Pattern Recognition*, 37(6):1245–1265, jun 2004. ISSN 00313203. doi: 10.1016/j.patcog.2003.12.005.

[299] Y. Huang and C. Suen. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):90–94, 1995. ISSN 01628828. doi: 10.1109/34.368145.

[300] A. Rahman and M. Fairhurst. An evaluation of multi-expert configurations for the recognition of handwritten numerals. *Pattern Recognition*, 31(9): 1255–1273, sep 1998. ISSN 00313203. doi: 10.1016/S0031-3203(97)00161-1.

[301] Y. Gal, J. Hron, and A. Kendall. Concrete Dropout. In *Advances in Neural Information Processing Systems*, volume 30, pages 3581–3590. Curran Associates, Inc., 2017.

[302] E. Hüllermeier and W. Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110(3):457–506, mar 2021. ISSN 0885-6125. doi: 10.1007/s10994-021-05946-3.

[303] R. Senge, S. Bösner, K. Dembczyński, J. Haasenritter, O. Hirsch, N. Donner-Banzhoff, and E. Hüllermeier. Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty. *Information Sciences*, 255:16–29, jan 2014. ISSN 00200255. doi: 10.1016/j.ins.2013.07.030.

[304] N. R. Prasad, S. Almanza-Garcia, and T. T. Lu. Anomaly detection. *Computers, Materials and Continua*, 14(1):1–22, 2009. ISSN 15462218. doi: 10.1145/1541880.1541882.

[305] D. Ulmer and G. Cinà. Know Your Limits: Uncertainty Estimation with ReLU Classifiers Fails at Reliable OOD Detection. *arXiv preprint arXiv:2012.05329*, dec 2020. ISSN 23318422.

[306] M. Markou and S. Singh. Novelty detection: a review—part 1: statistical approaches. *Signal Processing*, 83(12):2481–2497, dec 2003. ISSN 01651684. doi: 10.1016/j.sigpro.2003.07.018.

[307] W. Liu, X. Wang, J. Owens, and Y. Li. Energy-based out-of-distribution detection. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21464–21475. Curran Associates, Inc., 2020.

[308] L. Perini, V. Vercruyssen, and J. Davis. Quantifying the Confidence of Anomaly Detectors in Their Example-Wise Predictions. In *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Ghent, Belgium, 2020. ECML-PKDD.

[309] L. Perini, C. Galvin, and V. Vercruyssen. A Ranking Stability Measure for Quantifying the Robustness of Anomaly Detection Methods. *Communications in Computer and Information Science*, 1323:397–408, 2020. ISSN 18650937. doi: 10.1007/978-3-030-65965-3_27.

[310] B. Settles. Active Learning Literature Survey, 2009.

[311] Y. Fu, X. Zhu, and B. Li. A survey on instance selection for active learning. *Knowledge and Information Systems*, 35(2):249–283, may 2013. ISSN 0219-1377. doi: 10.1007/s10115-012-0507-8.

[312] C. Zhang and K. Chaudhuri. Beyond disagreement-based agnostic active learning. *Advances in Neural Information Processing Systems*, 1(January): 442–450, 2014. ISSN 10495258.

[313] V.-L. Nguyen, S. Destercke, and E. Hüllermeier. Epistemic Uncertainty Sampling. In *International Conference On Discovery Science DS 2019*, volume 11828 LNAI, pages 72–86. Springer, Cham, 2019. ISBN 9783030337773. doi: 10.1007/978-3-030-33778-0_7.

[314] L. 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)*, pages 2334–2343. IEEE, dec 2019. ISBN 978-1-7281-0858-2. doi: 10.1109/BigData47090.2019.9006453.

[315] 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 (ADMA2021)*, pages 1–15, Sydney, Australia, 2021. Springer Verlag.

[316] A. Rupp, A. Masieri, and T. Dornbusch. Durability Transfer Concept for the Monitoring of the Load and Stress Conditions on Vehicles. In *Innovative Automotive Technology IAT*, pages 31–39, Bled, Slovenia, 2005.

[317] M. Städele, A. Rupp, G. Willén, P. Romann, and M. Streicher. Ableitung der Beanspruchungs- und Lastzeitfolgen aus Referenzgröβen bei Langzeitmessungen an Nutzfahrzeugen. In *41. Tagung des DVM-Arbeitskreises Betriebsfestigkeit*, Ingolstadt, Germany, 2014.

[318] S. Balakrishnan, A. PP, R. V. Kharul, and S. C. Accurate Estimation of Time Histories for Improved Durability Prediction Using Artificial Neural Networks. In *SAE Technical Paper*. SAE International, apr 2012. doi: 10.4271/2012-01-0023.

[319] H. Luo, M. Huang, and W. Xiong. Application of a Recurrent Neural Network and Simplified Semianalytical Method for Continuous Strain Histories Estimation. *Shock and Vibration*, 2019:7289314, 2019. ISSN 1070-9622. doi: 10.1155/2019/7289314.

[320] H. Ludanek and G. Bremer. Herausforderungen an die Betriebsfestigkeit zukünftiger Automobilentwicklungen. In *Tagung des DVM-Arbeitskreises Betriebsfestigkeit*, Wolfsburg, Germany, 2007.

[321] B. Hammer and A. Klem. Fahrzeugauslegung als Spagat zwischen Leichtbau und Produkthaftung. In *Tagung des DVM-Arbeitskreises Betriebsfestigkeit*, Ingolstadt, Germany, 2014.

[322] J. Dodds Colin. *Structural testing of complete vehicles, aggregates and components in the laboratory, The Test Engineer's Handbook*. Dodds & Associates, Edinburgh, Scotland, second edi edition, 2012.

[323] A. Halfpenny. Methods for accelerating dynamic durability tests. In *Proceedings of the Ninth International Conference on Recent Advances in Structural Dynamics*, Southampton, UK, 2006. University of Southampton.

[324] P. K. P, P. J, and K. Palanisamy. Optimization of Proving Ground Durability Test Sequence Based on Relative Damage Spectrum. In *SAE Technical Paper*. SAE International, 2018. doi: 10.4271/2018-01-0101.

[325] M. W. Sayers and S. M. Karamihas. *The little book of profiling*. University of Michigan, Ann Arbor, Transportation Research Institute, Michigan, 1998.

[326] G. Alessandroni, L. C. Klopfenstein, S. Delpriori, M. Dromedari, G. Luchetti, B. D. Paolini, A. Seraghiti, E. Lattanzi, V. Freschi, A. Carini, and A. Bogliolo. SmartRoadSense: Collaborative road surface condition monitoring. In *UBICOMM 2014 - 8th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages 210–215, Rome, Italy, 2014. ISBN 9781612083537. doi: 10.13140/RG.2.1.3124.2726.

[327] F. Pedregosa, R. Weiss, M. Brucher, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[328] Google. Street View, 2020.

[329] D. L. Nuñez and M. Borsato. OntoProg: An ontology-based model for implementing Prognostics Health Management in mechanical machines. *Advanced Engineering Informatics*, 38(October):746–759, oct 2018. ISSN 14740346. doi: 10.1016/j.aei.2018.10.006.

[330] S. Al-Dahidi, F. Di Maio, P. Baraldi, and E. Zio. Remaining useful life estimation in heterogeneous fleets working under variable operating conditions. *Reliability Engineering & System Safety*, 156:109–124, dec 2016. ISSN 09518320. doi: 10.1016/j.ress.2016.07.019.

[331] X. Chesterman, T. Verstraeten, P.-J. Daems, and J. Helsen. Condition monitoring of wind turbines using machine learning based anomaly detection and statistical techniques for the extraction of 'healthy data'. *Annual Conference of the PHM Society*, 13(1):1–12, nov 2021. ISSN 2325-0178. doi: 10.36001/phmconf.2021.v13i1.2980.

[332] A. D. Brabandere, T. Op, K. Hendrickx, and W. Meert. TSFuse : Automated feature construction for multiple time series data. *Machine Learning*, page 54, 2021.

[333] B. Cornelis, K. Hendrickx, D. Hoffmann, M. Horst, M. Vollbrecht, and M. Kienert. Customer-correlated drive cycle generation for realistic fuel consumption and driving emissions testing. In *JSAE Annual Congres 2019*, Yokohama, Japan, 2019.

Curriculum vitae

Kilian Hendrickx was born in Brasschaat on April 8th 1992 and received secondary education from Koninklijk Atheneum Kapellen. He received a Bachelor of Science and Master of Science in Information Engineering Technology from Ghent University, respectively in 2014 and 2015. In 2016, he obtained a Master of Science in Artificial Intelligence from KU Leuven. His master theses were entitled "Advanced analysis of dynamic position measurements for bike fitting" and "A data-driven approach for plantar pressure analysis".

Kilian started his doctoral studies in the DTAI research group in 2016 in close collaboration with Siemens Digital Industries Software (Leuven). He was supervised by Prof. Dr. Jesse Davis and co-supervised by Prof. Dr. Ir. Konstantinos Gryllias (KU Leuven, LMSD, Department of Mechanical Engineering) and Dr. Bram Cornelis (Siemens Digital Industries Software). In 2017, he acquired a personal 4-year Baekeland mandate from the Flemish Agency for Innovation and Entrepreneurship (VLAIO) which financially supported his work.

List of publications

6.3 Journal papers

- K. Hendrickx, W. Meert, Y. Mollet, J. Gyselinck, B. Cornelis, K. Gryllias, and J. Davis. A general anomaly detection framework for fleet-based condition monitoring of machines. *Mechanical Systems and Signal Processing*, 139:106585, may 2020. ISSN 08883270. doi: 10.1016/j.ymssp.2019.106585
- A. D. Brabandere, T. Op, K. Hendrickx, and W. Meert. TSFuse : Automated feature construction for multiple time series data. *Machine Learning*, page 54, 2021

6.4 Conference papers

- A. Mauricio, C. Freitas, J. Cuenca, K. Gryllias, B. Cornelis, K. Janssens, and K. Hendrickx. Condition monitoring of gears under medium rotational speed. In *24th Internation Congress on Sound and Vibration*, page 8, London, 2017
- B. Cornelis, K. Hendrickx, D. Hoffmann, M. Horst, M. Vollbrecht, and M. Kienert. Customer-correlated drive cycle generation for realistic fuel consumption and driving emissions testing. In *JSAE Annual Congres 2019*, Yokohama, Japan, 2019
- K. Hendrickx, W. Meert, B. Cornelis, K. Janssens, K. Gryllias, and J. Davis. A Fleet-Wide Approach for Condition Monitoring of Similar Machines Using Time-Series Clustering. In Springer, editor, *Proceedings of the 6th International Conference on Condition Monitoring of Machinery in Non-Stationary Operations, CMMNO'2018*, pages 101–110. Springer, Cham, Santander, Spain, jun 2019. doi: 10.1007/978-3-030-11220-2_11

- K. Hendrickx, W. Meert, B. Cornelis, K. Gryllias, and J. Davis. Similarity-based anomaly score for fleet-based condition monitoring. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society, PHM*. PHM Society, 2020
- K. Hendrickx, W. Meert, J. P. D. C. Patricio, B. Cornelis, K. Gryllias, and J. Davis. Fleet-based health monitoring for end-of-production-line and operational testing. In *Online Proceedings of ISMA2020*, Leuven, Belgium, 2020. KU Leuven
- 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 (ADMA2021)*, pages 1–15, Sydney, Australia, 2021. Springer Verlag

6.5 Papers under review

- K. Hendrickx, L. Perini, D. Van der Plas, W. Meert, and J. Davis. Machine Learning with a Reject Option: A survey. 2021



FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
DECLARATIVE LANGUAGES AND ARTIFICIAL INTELLIGENCE
Celestijnenlaan 200A box 2402
B-3001 Leuven
kilian.hendrickx@cs.kuleuven.be
http://www.dtai.cs.kuleuven.be