TOLERANT MACHINE LEARNING FOR DEFICIENT TRAINING DATA

A THESIS SUBMITTED TO AUCKLAND UNIVERSITY OF TECHNOLOGY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Supervisors

Professor Edmund M-K Lai

Associate Professor Roopak Sinha

Professor Muhammad Asif Naeem

November 2022

By

Benjamin James Denham

School of Engineering, Computer and Mathematical Sciences

Abstract

While supervised machine learning methods have shown great potential for automating time-consuming manual data classification tasks, their application is often hindered by deficiencies in training datasets that degrade model accuracy. This thesis proposes *tolerant machine learning* methods that can be applied despite common training data deficiencies. By enabling users to derive value proportional to data quality, tolerant machine learning makes exploring new machine learning applications more cost-effective.

The first training data deficiency addressed by this thesis is a lack of class labels for training data, including when the set of possible classes is unknown. Prior work in the weak supervision paradigm of *data programming* has sought to provide large quantities of data labels through user-defined heuristic labelling functions. Despite the development of methods to assist users in defining such functions, users must still have a small labelled dataset or at least upfront knowledge of the set of possible classes. The WITAN algorithm proposed in this thesis can suggest labelling functions without any initial supervision, supporting the user to discover classes progressively. Experiments with binary and multi-class datasets demonstrate WITAN's competitive efficiency and accuracy compared to alternative labelling methods, despite its lack of supervision.

The second training data deficiency addressed by this thesis is the problem of dataset shift, where the data distribution of the training dataset differs from that of a target population. Dataset shift is a challenging yet expected problem when estimating the prevalences of classes in different target samples — so-called *quantification*. Existing

quantification methods make strong assumptions on the nature of dataset shift that may not hold in practice. This thesis proposes a Gain-Some-Lose-Some (GSLS) quantification model that is experimentally demonstrated to provide more reliable quantification prediction intervals than existing methods under more general conditions of shift. GSLS is integrated into a decision tree for dynamically selecting an appropriate quantification method for a given target sample. Selection by a Kolmogorov-Smirnov test for any shift followed by a newly proposed "Adjusted Kolmogorov-Smirnov" test for non-prior shift is found to best balance quantification and runtime performance. Additionally, a framework is presented for constraining quantification prediction intervals to user-specified limits by requesting class labels from the user for smaller sets of instances than would be required with rejection of classifications based on confidence scores alone.

The third and final training data deficiency addressed by this thesis is the problem of class noise. When a class noise process distorts the relationships between input features and class labels or true class values, a classifier should reject instances for which it cannot provide a confident classification. This thesis demonstrates how the popular model-agnostic confidence-thresholding rejection method does not leverage relationships between input features and class noise. A novel model-agnostic null-labelling rejection method is proposed to learn such relationships, and an experimental evaluation demonstrates its ability to achieve a significantly better trade-off between classification error and the rate of rejection under an evaluation framework that unifies prior theories for combining rejecting-classifiers. Additionally, null-labelling is demonstrated to enable users to understand relationships between input features and regions of class noise, allowing them to identify and potentially address sources of noise.

Finally, a software architecture is presented that combines all of the presented methods for addressing training data deficiencies. The architecture demonstrates a tolerant machine learning system with minimal data quality requirements, allowing users to start deriving value from their data with less upfront effort.

Contents

Abstract **2**

Attestation of Authorship **9**

Publications **10**

Acknowledgements **11**

Intellectual Property Rights **12**

Confidential Material **13**

1 Introduction **14**
1.1 Motivation 14
1.2 Research Objectives 18
1.3 Contributions 22
1.4 Thesis Structure 24

2 Suggesting Labelling Functions without Supervision for Low-Effort Data Programming **26**
2.1 Introduction 26
2.2 Literature Review 30
2.2.1 Data Programming for Weak Supervision 30
2.2.2 Assisted Data Programming 32
2.3 The Proposed Algorithm 35
2.3.1 LF Utility Function 38
2.3.2 Core of the Algorithm 42
2.3.3 Conjunctive and Disjunctive LFs 44
2.3.4 Incorporating Feedback 47
2.3.5 Analysis of WITAN 48
2.4 Multi-class Extension of IWS 53
2.5 Experimental Study 55
2.5.1 Experiment Framework 55
2.5.2 Binary-Class Experiments 60
2.5.3 Runtime Study 66

2.5.4 Multi-Class Experiments 66
2.5.5 Ablation Study 68
2.6 Conclusion 70

3 Reliable Quantification with Constrained Error Under Unknown Dataset Shift 71
3.1 Introduction 71
3.2 Literature Review 76
3.2.1 Types of Dataset Shift 76
3.2.2 Quantification Methods 78
3.3 Proposed GSLS Model of General Shift 82
3.3.1 Fitting the GSLS Model 84
3.3.2 Limitations of the Minimal Shift Criterion 88
3.3.3 Prediction Intervals for GSLS 90
3.4 Dynamic Quantifier Selection 93
3.4.1 Testing for Any Dataset Shift 94
3.4.2 Testing for Non-Prior Dataset Shift 95
3.5 Constrained Quantification Intervals 97
3.5.1 Framework for Constrained Intervals 98
3.5.2 Constrained PCC Intervals 101
3.5.3 Constrained EM Intervals 102
3.5.4 Constrained GSLS Intervals 103
3.6 Experimental Study 108
3.6.1 Experiment Framework 109
3.6.2 Comparison of Quantification Methods 111
3.6.3 Analysis of Shift Misestimation 117
3.6.4 Histogram Bins Sensitivity Analysis 121
3.6.5 Shift Detection Experiments 123
3.6.6 Constrained Quantification Interval Experiments 126
3.6.7 Synthetic Example Experiments 130
3.7 Conclusion 135

4 Learning Regions of Classifications to Reject Under Class Noise 136
4.1 Introduction 136
4.2 Literature Review 142
4.2.1 Background on Class Noise 142
4.2.2 Related Work on Rejection 143
4.3 Foundations of Confidence-Thresholding Rejection 146
4.4 Proposed Null-Labelling Method for Rejection 148
4.5 Combining Sets of Rejecting-Classifiers 155
4.6 Experimental Study 161
4.6.1 Experiment Framework 161
4.6.2 Experiment Results 164
4.7 Conclusion 168

5 Integration of Components for a Tolerant Machine Learning Architecture 169
5.1 Introduction 169
5.2 Logical View of the Architecture 171
5.2.1 Classifier Training 174
5.2.2 Classifier Application 175
5.2.3 Variability Guide 176
5.3 Usage Scenarios 176
5.4 Conclusion 178

6 Conclusions and Future Directions 179
6.1 Limitations 180
6.2 Directions for Future Work 181
6.2.1 More Diverse Applications of Tolerant Machine Learning . . . 181
6.2.2 Improved Accuracy of Tolerant Machine Learning Methods . 182
6.2.3 Tolerant Machine Learning for Broader Deficiencies 182

References 184

Appendices 193

A Application for Embargo 193

List of Tables

2.1 Assisted Data Programming Approaches. 36
2.2 Binary and multi-class datasets used in experiments. 61
2.3 Binary classification F1 scores for unseeded and seeded labelling methods at 25 and 100 interactions (IC). 62
2.4 Runtime in seconds for labelling methods. 67
2.5 F1 scores of WITAN variants. 69

3.1 Benchmark datasets for quantification. 110
3.2 Coverage of true class proportions by 80% prediction intervals. 112
3.3 Mean (with std. dev.) quantification absolute error in percentage points. 115
3.4 Coverage (Cov.) and mean (with std. dev.) absolute error (AE) in percentage points on plankton datasets with real-world shift. 116
3.5 Mean (with std. dev.) 80% prediction interval widths in percentage points. PCC and EM are omitted, as their intervals frequently have inadequate coverage. Bold indicates when one of GSLS or TGSLS is significantly narrower than the other (by a corrected re-sampled t-test; $\alpha = 0.05$). 120
3.6 Mean (with std. dev.) of GSLS runtimes (in msec). 122
3.7 Frequency of shift tests detecting shift under varied shift conditions. . 124
3.8 Coverage (Cov.) and mean (with std. dev.) absolute error (AE) in percentage points for the KS+AKS dynamic quantification method. . . 126
3.9 Comparison of methods for constraining quantification intervals on benchmark datasets with varied shift conditions and plankton datasets with real-world shift. Statistics are given in percentage points and reported as means with (std. dev.) where applicable. 128
3.10 Comparison of method runtimes for constraining quantification intervals reported as median seconds with (std. dev.). 130
3.11 Coverage of true class proportions by 80% prediction intervals on synthetic example datasets. 133
3.12 Comparison of methods for constraining quantification intervals on synthetic example datasets. Statistics are given in percentage points and reported as means with (std. dev.) where applicable. 134

4.1 Benchmark datasets for classification with rejection. 162
4.2 Results for CT and NL on noisy datasets. 164

List of Figures

1.1 The concept of tolerant machine learning: delivering value despite training data deficiencies. 19
1.2 Structure of the chapters within this thesis. 24

2.1 Logical view of WITAN, demonstrating 1) initial unsupervised generation of LFs, 2) user review and assignment of class labels to LFs, and 3) extending an existing set of LFs. 29
2.2 Examples of WITAN-generated LF sets. 51
2.3 Critical difference diagrams for binary classification F1 of labelling methods at IC of 25 (above) and 100 (below). 64
2.4 Binary classification results. 65
2.5 Multi-class classification results. 68

3.1 Quantification prediction intervals under varied shift. 74
3.2 Gain-Some-Lose-Some (GSLS) Model. 83
3.3 Decision tree for selecting a quantification method. 94
3.4 The remaining weight tends to be overestimated when there is little shift, and underestimated under extreme shift. 118
3.5 Shift is underestimated for both ISX and ISP, but only the prediction intervals of ISP are negatively impacted. 119
3.6 Mean $\hat{w}^R$ for different bin and target instance counts. 122
3.7 Comparison of coverage and absolute error for quantification method selection based on different combinations of shift tests. 124

4.1 Comparison of confidence-thresholding (CT) and null-labelling (NL) on a dataset with not-at-random class noise. 139
4.2 Model of NNAR class noise. 143
4.3 Comparison of CT and NL-2 on the skin segmentation dataset. 151
4.4 Plotting and interpolation between classifiers. 160
4.5 Comparison of CT and NL on benchmark datasets. 166
4.6 Increasing NL coverage on letter-recognition. 167

5.1 Context diagram for the tolerant machine learning system. 170
5.2 Legend for logical view diagrams. 171
5.3 Logical view of components involved in classifier training. 172
5.4 Logical view of components involved in classification and quantification. 173

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.

Signature of candidate

Publications

Denham, B., Pears, R., & Naeem, M. A. (2020). Null-Labelling: A Generic Approach for Learning in the Presence of Class Noise. In *2020 IEEE International Conference on Data Mining (ICDM)* (pp. 990-995). IEEE.

Denham, B., Lai, E. M., Sinha, R., & Naeem, M. A. (2021). Gain-Some-Lose-Some: Reliable Quantification Under General Dataset Shift. In *2021 IEEE International Conference on Data Mining (ICDM)* (pp. 1048-1053). IEEE.

Denham, B., Lai, E. M., Sinha, R., & Naeem, M. A. (2022). Witan: unsupervised labelling function generation for assisted data programming. Proceedings of the VLDB Endowment, 15(11), 2334-2347.

Denham, B., Lai, E. M., Sinha, R., & Naeem, M. A. (2022). Dynamic Quantification with Constrained Error Under Unknown General Dataset Shift. Manuscript submitted for publication to *IEEE Transactions on Knowledge and Data Engineering*.

Acknowledgements

Thanks first of all to my supervisors Professor Edmund Lai, Associate Professor Roopak Sinha, and Professor Muhammad Asif Naeem, who provided continuous support and guidance over the course of this PhD. I greatly appreciate their insights and advice that have helped me grow as a researcher. Thanks also to my initial primary supervisor Associate Professor Russel Pears, who was unfortunately unable to continue working on the project, but was instrumental in helping me set the direction for this research. Thanks also to Asif and Russel for encouraging me to pursue this PhD and for laying the groundwork for the project at Fisher & Paykel.

Thanks to all of the staff at Fisher & Paykel Appliances who I have collaborated with as part of this research. Thanks to the engineers whose extensive prior work in fault coding and willingness to share their knowledge provided the foundations for the application of this research. A special thanks to my primary collaborator at F&P, Daniel Lee, whose experience and ideas were invaluable in guiding this research to address real and challenging problems in the application of machine learning. Thanks also to my past and present colleagues at DataMasque who have shared this journey with me, and especially to my manager Aimee Lin for her support.

Thanks to Callaghan Innovation for funding this research under an R&D Fellowship Grant (FPAP1902), and to AUT for funding me through a Doctoral Fees Scholarship. Thanks also to Andy Hilton from AUT and Allen Guinibert from Fisher & Paykel for helping to arrange this funding. Thanks also to the AUT Ethics Committee for approving this research project (reference number 20/12, approved 26th March 2020).

Thanks to my mother Shona, father Paul, family, and friends who have encouraged me throughout my studies. A special thanks to Dr Grant Paton-Simpson for his constant encouragement, enlightening perspectives, and many phone calls about Python and much else. My most sincere thanks go to my mother Shona, who has always been there to support and encourage me, and has heard more about machine learning over the past years than she probably ever wanted to.

Finally, I give thanks to my God, who has given me the strength to complete this PhD and to whose glory I have performed this research.

Intellectual Property Rights

This research was funded by Callaghan Innovation R&D Fellowship Grant FPAP1902 for a Fisher & Paykel Appliances Limited project. In order to fulfil the obligations for this funding, the research was governed by an agreement between the Candidate (Benjamin Denham), Fisher & Paykel Appliances Limited, and Auckland University of Technology (AUT).

Confidential Material

In order to fulfil the obligations of the research agreement between the Candidate (Benjamin Denham), Fisher & Paykel Appliances Limited, and Auckland University of Technology (AUT), this thesis is to be embargoed for a period of 12 months from its lodgement. The completed Application for Embargo Form (PGR16) is included in Appendix A.

Chapter 1

Introduction

This thesis presents novel methods for a proposed *tolerant machine learning* paradigm, seeking to enable users to derive value from supervised machine learning despite deficiencies in training data. The following sections motivate the need for tolerant machine learning, identify specific research objectives and contributions to address them, and outline the structure of the thesis.

1.1 Motivation

From monitoring plankton diversity through image classification (González et al., 2019) to identifying damage reports on social media during a disaster (Mouzannar, Rizk & Awad, 2018), supervised machine learning methods for classification have become an invaluable tool for automating tasks that would historically require time-consuming manual inspection of individual data instances. Such automation can provide more timely results to users and allow analyses to be scaled up from small samples to full populations. In some cases, automation may even open up the potential for applications with datasets that might otherwise go underutilised, such as when categorising customer issues from incident call logs (Forman, Kirshenbaum & Suermondt, 2006).

However, training an accurate supervised classifier requires a suitable training dataset. While state-of-the-art deep learning models have produced impressive results, they rely on increasingly large datasets for effective training (Marcus, 2018). Unfortunately, lacking data of sufficient quality is a common roadblock to adopting machine learning in practice (Loukides, 2022). While recent advances like transfer learning from large pre-trained models can help by reducing the amount of training data needed in domains where generalisation across tasks is possible (Zhuang et al., 2020), they do not obviate the need for task-specific training data that is adequately representative of the target problem and data population. For example, the correct categorisation for the description of a product fault may heavily depend on domain-specific terminology, including component or part names and error codes. In situations like these, the user's unique dataset may possess substantial value for addressing the problem, despite its deficiencies.

In order to illustrate concrete examples of training data deficiencies that are faced in practice, consider the application of supervised machine learning to measure the class frequencies of different types of faults in textual descriptions of warranty claims for whiteware appliances. As resolving warranty claims is a significant expense for an appliance manufacturer, a manufacturer may seek to reduce the number of warranty claims by introducing initiatives to improve quality during product design and manufacturing. In order to determine how they should prioritise their efforts, the manufacturer requires visibility of the prevalences of different fault types that lead to warranty claims. Textual descriptions of warranty claims will often be sourceable from call centre logs or notes as well as comments written by service technicians about each warranty claim they handle. Without machine learning, each warranty claim may be labelled with one of a set of fault classes either by having a domain expert manually read each claim's description or by applying hand-crafted rules that look for the presence of certain combinations of keywords. By instead training and applying a supervised classification model, the aim

is to classify the claims with less effort than fully manual labelling and with greater accuracy than simple keyword-matching rules.

When seeking to apply machine learning to automate the classification of an unlabelled dataset, it is not unrealistic to expect that acquiring class labels for a number of instances sufficient for model training may itself be prohibitively expensive or time-consuming. While an appliance manufacturer might be able to justify assigning a domain expert to manually label a few hundred or thousand warranty claims for a handful of product types, it is unlikely to be able to afford to scale that approach across its full range of product types with their unique characteristics and fault types. Furthermore, the domain expert may not even know the full set of possible fault types without reviewing a substantial proportion of warranty claims. This scenario illustrates the first training data deficiency that hinders machine learning deployment: a lack of class labels, including a lack of a complete set of possible classes.

Even when labelled data is available for model training, there may be significant differences between the distribution of the training data and the distribution of the target data to which the model will be applied. Consider applying a classifier to measure the frequencies of different appliance fault types over time; the need to continuously monitor fault rates implies an expectation for the data distribution to evolve, diverging from the distribution of an initial training sample. Such *dataset shift* (formally defined in Section 3.2.1) breaks the typical modelling assumption that training and target data samples are independently and identically distributed (i.i.d.), often resulting in a negative impact on model performance (Rabanser, Günnemann & Lipton, 2019; Takahashi & Braga, 2020; Lemberger & Panico, 2020). In cases of continuous shift over time (aka *drift*), it may be infeasible to guarantee an i.i.d. training dataset, as doing so would require continually acquiring ground truth labels for significant samples of new data. As dataset shift is often impractical to avoid and can result in untrustworthy models, it is the second data deficiency of interest that hinders machine learning deployment.

Even if large quantities of labelled training data that are representative of the target population can be acquired, it may still not be possible to train a reliable classifier. A human labeller may provide inconsistent ground-truth labels for instances they find difficult to classify, or worse, there may be some instances where the input features do not provide sufficient information to determine the correct classification. For example, the comments provided for a particular warranty claim may not provide enough detail to identify the type of fault definitively. This phenomenon, where a random process distorts the relationship between the input features and true class value or class label, is referred to as *class noise* (Frénay & Verleysen, 2013; formally defined in Section 4.2.1). Even an ideal classifier may have a high rate of errors on instances affected by class noise, making it too unreliable for users to trust. Because class noise may make it impossible to train a reliable classifier, it is the third data deficiency of interest that hinders machine learning deployment.

To summarise, the following training data deficiencies have been identified as significant hindrances to accurate and reliable machine learning:

1. A training dataset with an insufficient quantity or complete absence of class labels, including an incomplete or completely absent set of possible classes.

2. A training dataset with a data distribution that may not be representative of one or more intended target populations (i.e. the problem of *dataset shift*).

3. A training dataset where the class labels or the true class values are partially dependent on an unknown random process (i.e. the problem of *class noise*).

If such training data deficiencies degrade a classifier's performance to the point where it cannot meet the user's expectations of accuracy and reliability, then the user cannot rely on its outputs in a practical setting. In this way, training data deficiencies can prevent the user from gaining any value at all from applying machine learning to what data they

do have. This is a significant lost opportunity for users who have access to datasets that, while suffering from deficiencies, contain valuable information relevant to their purposes.

1.2 Research Objectives

It is the basis of this thesis that users of machine learning with deficient training datasets should still be able to derive a degree of value that is proportional to the quality of the training data; the partial automation of a time-consuming manual task is still better than no automation at all. Achieving this requires the development of machine learning methods that are tolerant of training data deficiencies, which is the primary research goal of this thesis:

> **Primary research goal:** Develop *tolerant machine learning* methods that enable users to confidently deploy and derive value from supervised classification models despite training data deficiencies.

Fig. 1.1 illustrates how tolerant machine learning could effectively remove the initial barrier of needing to construct a training dataset of sufficient quality before machine learning can deliver any value. By removing the need for a large upfront investment in data quality, tolerant machine learning would enable users to explore new applications more readily.

While prior research has sought to address the three key training data deficiencies identified in Section 1.1, existing state-of-the-art methods have shortcomings that limit their suitability to be deployed and relied upon in a tolerant machine learning system.

Recent work in the field of *weak supervision* has sought to address a lack of class labels by leveraging sources of labels that are less reliable but cheaper to acquire than traditional ground truth labels (J. Zhang, Hsieh, Yu, Zhang & Ratner, 2022). The

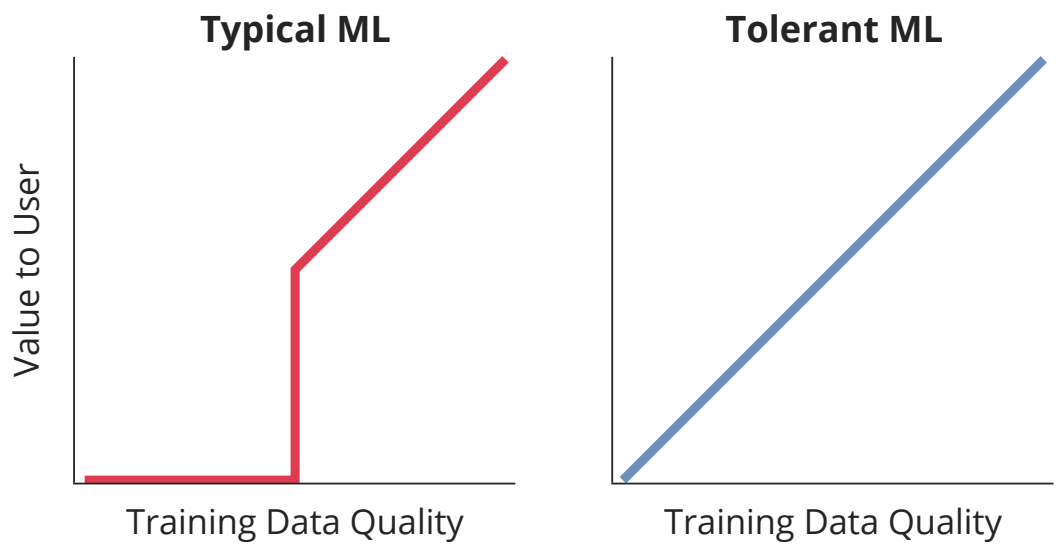


Figure 1.1: The concept of tolerant machine learning: delivering value despite training data deficiencies.

data programming paradigm has become particularly popular, allowing users to define heuristic *labelling functions* that assign labels based on simple conditions of input features (A. J. Ratner, De Sa, Wu, Selsam & Ré, 2016). For example, a labelling function may be defined to assign a `noise fault` class to any warranty claim with a description containing the word "`loud`". However, designing effective labelling functions still requires non-trivial effort and understanding of the classification problem on the part of the user. *Assisted data programming* methods (reviewed in Section 2.2.2) have attempted to help users design labelling functions. However, all existing methods still make non-trivial demands of the user. They either require some initial ground-truth labels or assume that the user knows the full set of possible classes upfront. While it may seem like a trivial requirement for the user to know the possible classes, this is not the case for all classification tasks. A domain expert tasked with categorising warranty claims may not know the types of faults occurring until they have reviewed a substantial proportion, if not all, of the claims. This potential lack of a complete set of possible classes is an important consideration not sufficiently addressed by existing methods.

The field of *quantification* has sought to provide more reliable estimates of class proportions in target samples under dataset shift than can be achieved by simply counting the outputs of a classifier (González, Castaño, Chawla & Coz, 2017). Several quantification methods can even provide prediction intervals for class proportions, allowing the user to understand the degree to which they may rely on the accuracy of estimates under the observed dataset shift. However, many quantification methods rely on strong assumptions about the nature of shift (González et al., 2017; also reviewed in Section 3.2). Not only may such assumptions not hold in practice, but it may not even be known which assumptions will or will not hold in a given sample of target data. When their assumptions do not hold, quantification methods may produce inaccurate estimates with prediction intervals that are misleadingly narrow. Furthermore, while prediction intervals communicate the degree of potential error to the user, they do not provide any mechanism for the user to sufficiently improve estimate accuracy in cases where excessive uncertainty makes estimates unusable.

The field of *classification with rejection* (Chow, 1970) has sought to enable the deployment of classifiers even when a confident classification cannot be produced for every target instance, such as in cases of class noise. A *rejecting-classifier* learns to *reject* instances for which it cannot provide a reliable prediction so that the user may provide a class label (if possible) or otherwise handle them manually. While rejection requires additional manual effort from the user, it can enable the deployment of a classifier for at least partial automation of a task in circumstances where it is impossible to provide a sufficiently confident classification for every instance. However, existing approaches to rejection have limitations that can make them suboptimal for addressing class noise (reviewed in Section 4.2.2). Many rejection methods rely on classification confidence scores, which are not always ideal discriminators of class noise when the relationship between input features and regions of class noise is not purely random (as demonstrated in Section 4.1). Other methods (reviewed in Section 4.2.2) rely on custom

classifier and loss function implementations that cannot be easily adapted to different classification models. Finally, existing methods do not provide a natural means for the user to understand which features identify rejected instances, which could help the user identify and potentially address sources of class noise.

These research gaps motivate the objectives of this thesis, which are focused on the development of tolerant machine learning methods to address the three key training data deficiencies identified in Section 1.1:

RO1 Develop an assisted data programming method to enable users to efficiently construct sets of labelling functions when no instance labels or even a set of classes are provided upfront, while still achieving classification performance competitive with or exceeding existing methods.

RO2 Develop a methodology for reliable quantification under unknown dataset shift with constrainable prediction intervals, achieved by the following sub-objectives:

- **RO2.1** Develop a quantification method that can be safely applied under weaker assumptions of dataset shift than existing quantification methods.
- **RO2.2** Develop an approach for dynamically selecting an appropriate quantification method for a given target sample in order to exceed the performance of a single, static quantification method.
- **RO2.3** Develop a framework for constraining quantification prediction intervals to user-specified limits by reducing uncertainty through requests for ground truth labels from the user for minimal sets of target instances.

RO3 Develop a model-agnostic rejection method to identify and reject regions of class noise in order to achieve a better trade-off between classification error and rate of rejection than confidence-based rejection.

1.3 Contributions

In order to achieve the research objectives established to address specific training data deficiencies, this thesis makes the following contributions of novel methods for tolerant machine learning:

1. The deficiency of a lack of class labels and set of possible classes (RO1) is addressed by the following contributions made in Chapter 2:
   (a) The novel WITAN algorithm for assisted data programming, which enables users to discover classes progressively by proposing labelling functions without any initial supervision.
   (b) Optional extensions to WITAN that enable it to construct labelling functions based on conjunctions and disjunctions of feature conditions and to guide the selection of additional functions based on user approval of previously generated functions, seed functions, or both.
   (c) An experimental evaluation of WITAN against other labelling function generation and labelling approaches for both binary and multi-class classification tasks to demonstrate the ability of WITAN to approach peak classification performance with less user effort than other methods.
      i. The previously proposed Interactive Weak Supervision assisted data programming method (Boecking, Neiswanger, Xing & Dubrawski, 2021) is extended from the binary to the multi-class case for inclusion in the evaluation.
2. The deficiency of dataset shift (RO2) is addressed by the following contributions made in Chapter 3:
   (a) A novel *Gain-Some-Lose-Some* (GSLS) model for reliable quantification under general dataset shift.

(b) A decision tree for selecting an appropriate quantification method for given source and target samples based on previously proposed statistical tests for detecting shift and a novel proposal for efficiently detecting non-prior shift.

(c) A framework based on binary integer programming for constraining quantification intervals to user-specified limits through requests for minimal target instance class labels from the user.

(d) An experimental comparison of these methods to the state-of-the-art, using benchmark datasets under varied conditions of shift and a real-world dataset previously used to study dataset shift.

3. The deficiency of class noise (RO3) is addressed by the following contributions made in Chapter 4:

(a) A novel model-agnostic *null-labelling* method for rejection that explicitly learns to reject regions of class noise.

(b) Interpretation of null-labelled models for identifying features that are correlated with class noise.

(c) A unification of prior theories to produce a framework for combining rejecting-classifiers and evaluating performance across different rejection rates.

(d) An experimental evaluation demonstrating that *null-labelling* can achieve a better tradeoff between classification error rates and rejection rates than confidence-thresholding in the presence of class noise.

4. Each of the novel methods for tolerant machine learning can be used as a component in an end-to-end machine learning system that consumes unlabelled data and specific training inputs from a domain expert in order to classify and quantify the data. Integrating the components produces a system that is tolerant to the full range of training data deficiencies identified above. Chapter 5 presents an

architecture for such a tolerant machine learning system, making the following contributions:

(a) A description of how a tolerant machine learning system should interact with external systems and users.

(b) Identification of the desired qualities for a tolerant machine learning system.

(c) The presentation of a logical view of the architecture, including points of variability within the architecture, in order to demonstrate the reliability and modifiability inherent in the architecture.

(d) A set of usage scenarios that demonstrate the flexibility of the architecture to support a variety of interaction modes and applications.

1.4 Thesis Structure

Fig. 1.2 presents the structure of the thesis chapters diagrammatically, and is followed by an outline of how the contributions established above are presented in each chapter.

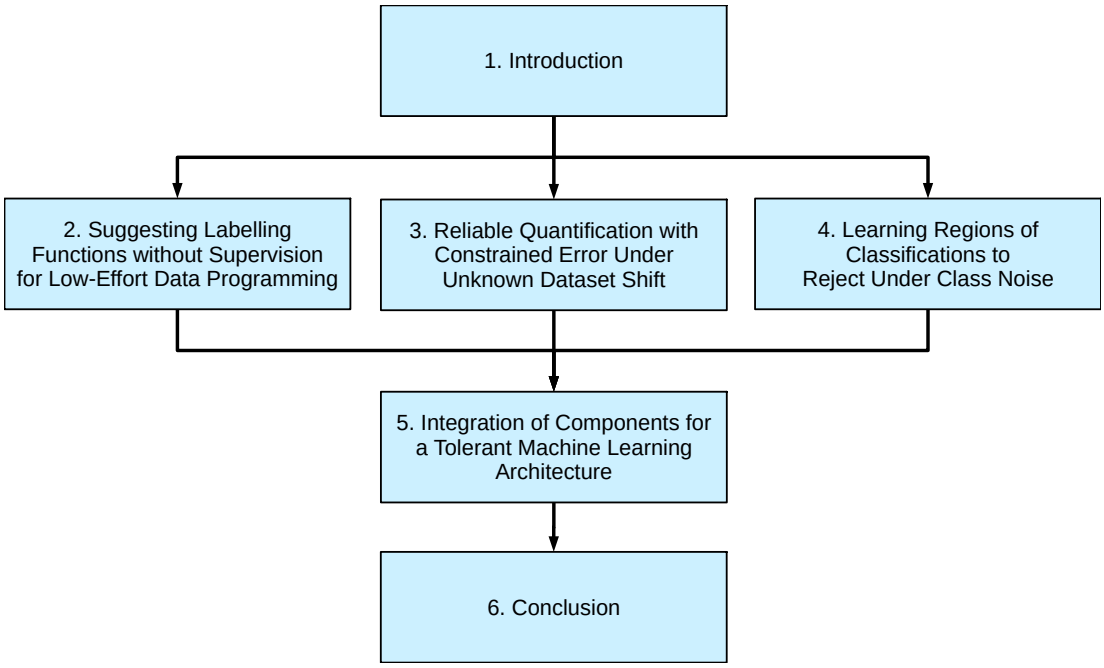


Figure 1.2: Structure of the chapters within this thesis.

- Chapters 2, 3,and 4 present research contributions that address each of the three research objectives identified above. Each chapter includes its own review of literature that is relevant to its research objective.
  - Chapter 2 addresses RO1 by presenting a novel algorithm for assisted data programming without any initial supervision. This chapter is based on the work presented in "Witan: unsupervised labelling function generation for assisted data programming" (Denham, Lai, Sinha & Naeem, 2022b).
  - Chapter 3 addresses RO2 by presenting a novel quantification method under general dataset shift, a decision tree for dynamic quantification method selection, and a framework for constraining quantification prediction intervals. This chapter is based on the work presented in "Gain-Some-Lose-Some: Reliable Quantification Under General Dataset Shift" (Denham, Lai, Sinha & Naeem, 2021) and "Dynamic Quantification with Constrained Error Under Unknown General Dataset Shift" (Denham, Lai, Sinha & Naeem, 2022a).
  - Chapter 4 addresses RO3 by presenting a novel, model-agnostic algorithm for learning interpretable regions of class noise for which classifications should be rejected. This chapter is based on the work presented in "Null-Labelling: A Generic Approach for Learning in the Presence of Class Noise" (Denham, Pears & Naeem, 2020).
- Chapter 5 presents the architecture for a tolerant machine learning system employing the methods developed in previous chapters.
- Chapter 6 provides concluding thoughts and directions for future research.

Chapter 2

Suggesting Labelling Functions without Supervision for Low-Effort Data Programming

2.1 Introduction

This chapter is based on the work presented in "Witan: unsupervised labelling function generation for assisted data programming" (Denham et al., 2022b), and addresses RO1:

RO1 Develop an assisted data programming method to enable users to efficiently construct sets of labelling functions when no instance labels or even a set of classes are provided upfront, while still achieving classification performance competitive with or exceeding existing methods.

The benefits of leveraging large datasets for supervised training of classifiers have been clearly demonstrated in recent years. In particular, deep learning algorithms are able to deliver impressive results in text classification and image recognition tasks but are well-known for their reliance on massive training datasets (Marcus, 2018).

Unfortunately, acquiring training labels in such large quantities is often prohibitively expensive, especially if labelling requires the assessment of training instances by a domain expert. The scarcity of training labels has led to the development of *weak supervision* methods that leverage supervision sources that are noisier but cheaper to acquire than traditional ground truth training labels. Such weak supervision sources include crowd-sourced labels (Rajchl et al., 2016), existing knowledge-bases (in so-called *distant supervision*; Alrashdi & O'Keefe, 2020), and even class names alone (Wang, Mekala & Shang, 2021).

Data programming has emerged as a popular weak supervision paradigm, accepting supervision in the form of user-defined heuristic *labelling functions* (LFs) that assign class labels to instances, typically based on conditions of instance features (A. J. Ratner et al., 2016). For example, an LF for text sentiment classification might label any instance containing the word "wonderful" with the "positive" class. Not only can data programming save effort by enabling the user to label potentially hundreds of instances with each LF, but LFs can also explicitly capture the user's domain knowledge. A set of LFs can serve as high-level documentation of labelling decisions in a way that a large set of labelled instances cannot. This also means that labelling decisions can be re-evaluated and changed in light of new requirements by simply updating the LFs. These attributes make data programming a natural paradigm for users to systematically manage and improve their training data in line with the recent trend towards data-centric AI, where an emphasis is placed on improving training data over models (Ng, 2021).

Despite these benefits, users have reported (Mallinar et al., 2019) the need for guidance in designing LFs that 1) are accurate, 2) cover many instances, and 3) are not overly biased towards certain classes. While a wide variety of *assisted data programming* approaches have been proposed to help users design LFs (see Section 2.2.2), almost all of these approaches assume that the user already knows at least the set of classes they wish to identify within their dataset. However, there are many situations where this will

not be the case; a user seeking to classify articles by topic or warranty claims by fault type may not have a fixed set of topics or faults in mind before exploring the dataset. Such users would benefit from an unsupervised analysis of their dataset to identify a variety of potential LFs from which they could choose. Even if the user has some idea of their desired classification scheme, such an unsupervised analysis could help them avoid human bias in manual LF design and identify other relevant concepts within the dataset, such as natural sub-classes of their target classes (e.g. subtopics or special cases of faults).

In order to address this need, this chapter proposes WITAN, an algorithm for unsupervised LF discovery. As illustrated in step 1 of Fig. 2.1, WITAN takes an unlabelled dataset and uses an information-theoretic approach to generate a set of potential LFs that are diverse not only in the instances they cover but also in the dimensions of the dataset they abstract. In step 2, a domain expert holistically reviews the entire set of generated LFs, selecting LFs relevant to their classification task and deciding which class labels those LFs should assign. Finally, the optional step 3 demonstrates how WITAN can extend an existing set of LFs with complementary LFs that increase diversity, whether the existing LFs were generated by WITAN or provided by the user as *seed* LFs. Analogous to the historical royal advisors for which it is named (Betham, 1834), WITAN aims to elicit new insights and assist in classifier training as much as possible while still empowering users with full control over the set of LFs — a user can easily interpret the behaviour of a proposed LF and decide whether or not to use it to assign a class label, and whether to manually amend the decision criteria to fit their classification task better. In addition to these benefits, an experimental evaluation demonstrates that WITAN-generated LFs result in competitive classification performance compared to alternative LF generation and instance labelling approaches.

The rest of this chapter is organised as follows. Section 2.2 presents the current state of the data programming approach to weak supervision with an emphasis on assisted data

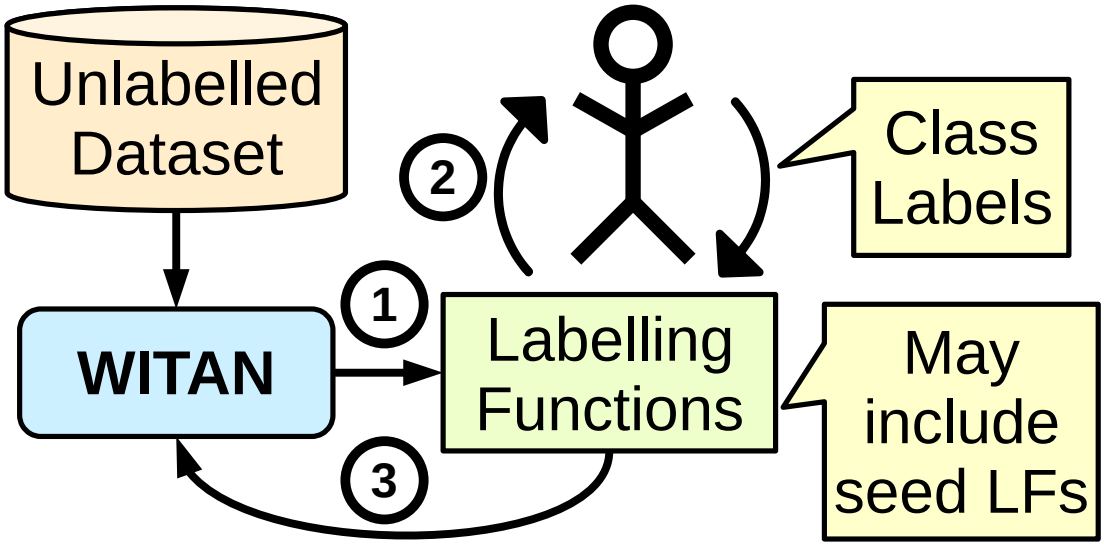


Figure 2.1: Logical view of WITAN, demonstrating 1) initial unsupervised generation of LFs, 2) user review and assignment of class labels to LFs, and 3) extending an existing set of LFs.

programming. The novel WITAN algorithm for unsupervised LF discovery is described in Section 2.3, which includes the core algorithm as well as optional components to construct LFs based on conjunctions and disjunctions of feature conditions and to guide the selection of additional LFs based on user approval of previously generated LFs, seed LFs, or both. Section 2.4 extends a previously proposed LF generation method known as Interactive Weak Supervision (IWS; Boecking et al., 2021) from the binary to the multi-class case for inclusion in an experimental evaluation of WITAN. In Section 2.5, results of extensive experimental evaluation of WITAN against other LF generation and labelling approaches for both binary and multi-class classification tasks are presented. They demonstrate the ability of WITAN to approach peak classification performance with less user effort than other methods. Finally, Section 2.6 concludes the chapter.

2.2 Literature Review

This section reviews the paradigm of data programming for weak supervision, focusing on methods for assisted data programming.

2.2.1 Data Programming for Weak Supervision

Data programming is a recently proposed paradigm for weak supervision from user-defined labelling functions (LFs; A. J. Ratner et al., 2016). In the context of a classification task to predict the class label $y \in \mathcal{Y}$ for an instance with features $x \in \mathcal{X}$, a labelling function λ uses an instance's features to either assign a predicted class label or abstain ($\varnothing$), such that $\lambda : \mathcal{X} \to \mathcal{Y} \cup \{\varnothing\}$ (A. Ratner et al., 2017). While LFs can potentially use complex decision criteria based on distantly supervising knowledge bases (A. Ratner et al., 2017) or arbitrary supervised classifiers (A. Ratner et al., 2017; Varma & Ré, 2018), this review focuses on LFs that can be expressed as conditions on instance features, such that a user would be able to interpret and potentially amend the decision criteria of a machine-generated LF.

As LFs essentially act as imperfect classifiers, one or more LFs may assign different class labels to a given instance. To account for this, arguably the most important component in a data programming system is a *labelling model* to aggregate LF outputs for training instances into a consistent set of training class labels. The seminal Snorkel system (A. Ratner et al., 2017) learns a generative model that estimates the accuracies of LFs based solely on their agreements and disagreements on training instances without needing any ground truth labels. Snorkel's generative model is a factor graph that considers LFs to be imperfect (or noisy) classifiers having different error rates, where the errors made by each LF are assumed to be uncorrelated with the errors made by other LFs unless some dependency between a pair of LFs is explicitly modelled. The factor graph is optimised to learn the error rate of each LF given only the labels

assigned by LFs to training set instances. The learned error rates of LFs are then used to weight their classifications and produce *probabilistic training labels* for the training set, where each of m possible class labels is assigned a numeric probability: $\tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_m), \tilde{y}_i \in [0, 1]$. These probabilistic training labels can then be used to train a classifier using a noise-aware loss function. Alternative labelling models have been subsequently proposed based on triplet methods (Fu et al., 2020) and game optimisation (Arachie & Huang, 2021). Given the popularity of Snorkel's open-source library and its use in previous studies on assisted data programming, Snorkel's generative model is used in experiments in Section 2.5 to evaluate LFs produced by WITAN and other assisted data programming methods in the context of classification.

It is important for WITAN and other assisted data programming methods that aid in the construction of LFs to consider the desiderata for LFs that produce accurate classifiers. The primary trade-off in LF design is between coverage and accuracy (Varma & Ré, 2018; Boecking et al., 2021); LFs that are general enough to apply to many instances risk misclassifying a significant proportion of them, while highly specific and accurate LFs may cover too few instances for the subsequently trained classifier to generalise to broader patterns. An ideal set of LFs optimises for both coverage and accuracy, taking advantage of the fact that labelling models are best suited to resolving a moderate degree of overlap and disagreement between LFs (A. Ratner et al., 2017). As labelling models often assume the class distribution can be specified a priori or estimated from LF outputs (A. Ratner et al., 2017; Fu et al., 2020), it is also important for LF coverage to be representative of the true class distribution and not unduly biased towards certain classes over others. In order to avoid catastrophic results for extremely biased LFs, experiments in Section 2.5 apply Snorkel with a uniform class prior. Finally, while Snorkel has mechanisms for explicitly modelling dependencies between LFs (e.g. when one LF "reinforces" another by identically labelling a strict subset of its covered instances; Varma, Sala, He, Ratner & Ré, 2019), it assumes independence

by default (A. Ratner et al., 2017). Methods have been developed for automatically learning dependencies (Varma et al., 2019), and other labelling models are designed to be robust to dependencies (Arachie & Huang, 2021). However, because empirical results suggest classifier accuracy can be degraded by modelling all but the strongest dependencies (Cachay, Boecking & Dubrawski, 2021), experiments in Section 2.5 are limited to applying Snorkel under the assumption that all LFs are independent.

2.2.2 Assisted Data Programming

A variety of methods have previously been proposed to help users construct LFs with minimal effort. This section reviews these *assisted data programming (ADP)* methods, focusing on the interaction modes they support and how they compare to WITAN.

Assisting Manual LF Creation

Some ADP methods simplify the user's task of defining decision criteria for LFs. For text classification, one approach is to have the user simply specify a list of keywords associated with each class (Karamanolakis, Hsu & Gravano, 2019; B. Lu, Ott, Cardie & Tsou, 2011; Grechkin, Poon & Howe, 2018). Such keyword lists can be filtered of "noisy" keywords through co-training (Karamanolakis et al., 2019), used to seed LDA topic models applied as classifiers (B. Lu et al., 2011), or even applied to non-text classification tasks by leveraging text descriptions for some training instances (Grechkin et al., 2018). Another approach for assisting users is to identify subsets of instances (Cohen-Wang, Mussmann, Ratner & Ré, 2019) or regions of the input space (Varma, Iter, De Sa & Ré, 2017) where existing LFs perform poorly so that the user may target those deficiencies with additional LFs. However, all of these methods still rely on users designing criteria for LFs themselves, which relies on them manually reviewing many instances to identify common patterns of interest.

Instance-based Methods

Another family of ADP methods generates LFs based on a small number of manually labelled instances. Some of these methods still rely on the user providing the criteria that influenced their labelling decisions in order to form the basis of LFs, whether by making a search query for instances to cover with an LF (Mallinar et al., 2019), marking important keywords in text (Evensen, Ge, Choi & Çağatay Demiralp, 2020), or providing natural language explanations for their decisions (Hancock et al., 2018). Other methods can generate LFs based solely on labelled instances. One such method (Mallinar, Shah, Ho, Ugrani & Gupta, 2020) generates LFs that propagate manual labelling to similar instances found using a text search engine, though such instance-based LFs lack interpretable decision criteria. Other methods generate LFs by training arbitrary classifiers on a small labelled training set (Varma & Ré, 2018; Nashaat, Ghosh, Miller & Quader, 2020). The Snuba system (Varma & Ré, 2018) can produce reasonably interpretable LFs when a simple classification model (such as a decision stump) is used to generate LFs. Because of this, Snuba is included as a representative of instance-based LF generation methods in the experimental evaluation (Section 2.5).

Supervised LF Generation

Rather than having the user label instances, some assisted data programming methods propose LFs and ask the user to review their decision criteria and then approve or reject each LF. Similar approaches for active learning predate data programming, where users review iteratively proposed classification rules to expand an initially small labelled training set (Du & Ling, 2010; Rashidi & Cook, 2011). Methods for data programming have been proposed that can learn LFs either for a single "positive" class (Giacometti & Soulet, 2017; Galhotra, Golshan & Tan, 2021) or for both classes in a binary classification task (Kartchner, Ren, Nakajima An, Zhang & Mitchell, 2020; Boecking et

al., 2021). As it is important for the user to be able to interpret and evaluate the decision criteria of a proposed LF, these methods have been used to propose LFs with decision criteria based on the presence of text keywords (Galhotra et al., 2021; Kartchner et al., 2020; Boecking et al., 2021), itemset patterns (Giacometti & Soulet, 2017), regular expressions and parse tree structures (Galhotra et al., 2021), and neighbourhoods of images (Boecking et al., 2021). Of these methods, only Interactive Weak Supervision (IWS; Boecking et al., 2021) is amenable to both multi-class classification (achieved through extensions proposed in Section 2.4) and non-text-based LFs. Therefore, IWS will be included in the experimental evaluation (Section 2.5) as a representative of methods requiring user review of generated LFs. A limitation of this family of methods is their reliance on continuous feedback from the user after each LF is proposed, as the feedback informs the selection of the next LF.

Unsupervised LF Discovery

A key limitation of all the ADP approaches discussed above is that they require the user to know the set of classes for their classification task a priori. The ability to begin data programming without a clear idea of the final class structure is invaluable for many tasks, such as in document topic classification, where the user is interested in first discovering the range of topics that are present. This need inspired the development of WITAN, which requires no initial supervision to generate a set of possible LFs for the user to review and assign class labels to. While unsupervised clustering has previously been used to create LFs (Luo & Hauskrecht, 2018, 2019; Alonso Doval, 2021), such methods only present a single possible clustering to the user. In contrast, WITAN proposes diverse LFs that represent a variety of orthogonal partitionings of the data. The inherent subjectivity of clustering and the importance of allowing users to choose between equally valid clusterings has been recognised in the *interactive clustering* literature (Bae et al., 2020), though the majority of interactive clustering

methods do not produce clusters with interpretable inclusion criteria that are stable between user interactions (Neubauer, Peres, Fantinato, Lu & Reijers, 2021) as desired for LFs. *Clustering by intent* (CBI; Forman, Nachlieli & Keshet, 2015; Bergner & Krempl, 2016) is an interactive clustering approach that produces interpretable rules similar to the LFs generated by WITAN, but it relies on the user providing labelled instances for at least one class at the outset.

Table 2.1 compares WITAN to other ADP approaches; WITAN's key distinguishing properties are summarised below:

- The user does not need to design LF decision criteria; the user only evaluates and labels WITAN-generated LFs.
- No manual instance labelling is required; all labels are assigned by LFs with interpretable decision criteria.
- LFs can be generated in an unsupervised fashion without a priori knowledge of the set of classes.
- The user can review the whole set of LFs after generation; WITAN does not need feedback after generating each LF.

2.3 The Proposed Algorithm

This section provides a detailed description of the proposed WITAN algorithm for unsupervised LF generation. The utility measure for selecting LFs is described and then applied to formulate the core WITAN algorithm. Extensions are also presented for generating LFs with conjunctive and disjunctive conditions and for incorporating user feedback into LF selection.

WITAN is assumed to be provided with a training set $X \in \{0,1\}^{n,m}$ of n unlabelled instances comprised of m binary features. Binary features are required by WITAN's

Table 2.1: Assisted Data Programming Approaches.

Does not require:	**Labelled instances**	**Designed LFs**	**Prior set of classes**	**Continuous feedback**
LF design aids (Karamanolakis et al., 2019; B. Lu et al., 2011; Grechkin et al., 2018; Cohen-Wang et al., 2019; Varma et al., 2017)	✓	×	×	×
Instance-based LF design aids (Mallinar et al., 2019; Evensen et al., 2020; Hancock et al., 2018)	×	×	×	✓
Instance-based LF generation (Mallinar et al., 2020; Varma & Ré, 2018; Nashaat et al., 2020)	×	✓	×	✓
Feedback-based LF generation (Giacometti & Soulet, 2017; Galhotra et al., 2021; Kartchner et al., 2020; Boecking et al., 2021)	✓	✓	×	×
LFs from clusters (Luo & Hauskrecht, 2018, 2019; Alonso Doval, 2021)	✓	User interaction not supported		
Clustering by intent (Forman et al., 2015; Bergner & Krempl, 2016)	Min 1 class	✓	✓	✓
WITAN	✓	✓	✓	✓

utility function, but they also allow constructing an LF λ using any Boolean combination of a feature subset $d^{\lambda} \in \mathcal{P}(\{1, \ldots, m\})$ to produce a coverage vector $c^{\lambda} \in \{0, 1\}^n$, such that λ assigns a user-given class label y^{λ} to instances covered by c^{λ} and abstains on non-covered instances:

$$\lambda(x_i) = \begin{cases} y^{\lambda} & \text{if } c_i^{\lambda} = 1 \\ \varnothing & \text{otherwise} \end{cases} \tag{2.1}$$

For example, a simple LF λ_j that covers instances for which feature j is positive would have $d^{\lambda_j} = \{j\}$ and $c^{\lambda_j} = x_{*j}$, where x_{*j} is the jth column vector in X. Note that, as an unsupervised algorithm, WITAN selects LFs solely on the evaluation of each LF's coverage vector c^{λ} and leaves class label y^{λ} for the user to decide later. Following previous advice on selecting LFs (A. Ratner et al., 2017; Boecking et al., 2021), users should be instructed to approve an LF for a class if they believe it will be more accurate than an LF that randomly assigns any class label.

Datasets containing non-binary features can still be used with WITAN by applying appropriate feature transformations, such as one-hot encoding categorical features and binning or thresholding numeric features. In the LF families framework of Boecking et al. (2021), binary features could be produced from any LF family where each LF assigns only a single class value. These include LF families that test for the presence of a keyword in text, test for the presence of motifs in a times series, or detect the presence of objects in an image. Whatever binary features are used, it is important that a user should be able to easily understand which instances a feature would be positive for so that they can judge whether or not that feature would be a useful decision criterion for an LF.

Throughout the rest of this section, examples will be used to illustrate WITAN's behaviour. These examples are based on the context of applying WITAN to a set of binary bag-of-words features that each indicate the presence of a particular word in

a film or television review, with the intent of generating LFs for positive/negative sentiment classification.

2.3.1 LF Utility Function

In order for WITAN to select LFs to propose to the user, a utility function is required to compare LFs in a candidate set. Ideally, this function should select LFs that are good classifiers for one of the classes in the target classification task. A similar problem is faced by the "covering rule" learning algorithms used for classification, which must select feature conditions that optimise a rule's classification performance (Fürnkranz & Flach, 2005). To solve this problem, the classic CN2 rule learner (Clark & Niblett, 1989) takes an information-theoretic approach by preferring a rule with coverage vector c that minimises the conditional entropy $H(Y|c)$ of the class variable $Y \in \{y_1, \ldots, y_k\}$:

$$H(Y|c) = -\sum_{l=1}^{k} P(y_l|c) \log_2 P(y_l|c) \tag{2.2}$$

where $P(y_l|c)$ is the probability that $Y = y_l$ for instances covered by c, as estimated from the training set. Decision tree learners also commonly make branching decisions b that maximise the reduction in conditional entropy from the previous model state to the new state, referred to as the *information gain* (Fürnkranz & Flach, 2005; Quinlan, 1986): $IG(Y, b) = H(Y) - H(Y|b)$.

In the context of unsupervised LF generation, the immediate barrier to using entropy in a utility function is that the class labels Y for the training set are unknown, so they cannot be used to determine which LF conditions will help predict Y. However, assuming the binary features in X are relevant to the classification task, it is fair to assume that at least some features will correlate with Y. Therefore, WITAN will aim to construct an LF condition c^λ to predict a class value $y \in Y$ with $P(y|c^\lambda) \gg P(y)$ by reducing the conditional entropy of y-correlated features x_{*j}. To see how this

works, consider the following relationship between conditional probabilities $P(y|c^\lambda)$ and $P(y|x_{*j})$:

$$
\begin{aligned}
P(y|c^\lambda) &= P(x_{*j}|c^\lambda)P(y|c^\lambda, x_{*j}) + P(\neg x_{*j}|c^\lambda)P(y|c^\lambda, \neg x_{*j}) \\
\therefore P(y|c^\lambda) &\geq P(x_{*j}|c^\lambda)P(y|c^\lambda, x_{*j}) \text{ (law of total prob.)} \\
\text{and } P(y|c^\lambda) &\geq P(\neg x_{*j}|c^\lambda)P(y|c^\lambda, \neg x_{*j}) \text{ (law of total prob.)} \\
\\
P(y|c^\lambda, x_{*j}) &\to P(y|x_{*j}) \text{ as } P(c^\lambda|x_{*j}) \to 1 \text{ (prob. chain rule)} \\
P(c^\lambda|x_{*j}) &= P(x_{*j}|c^\lambda)P(c^\lambda)/P(x_{*j}) \text{ (Bayes' rule)} \\
\\
\therefore \min(P(y|c^\lambda)) &\to P(y|x_{*j}) \text{ as } P(x_{*j}|c^\lambda) \to 1 \;\&\; P(c^\lambda)/P(x_{*j}) \to 1 \\
\text{and } \min(P(y|c^\lambda)) &\to P(y|\neg x_{*j}) \text{ as } P(\neg x_{*j}|c^\lambda) \to 1 \;\&\; P(c^\lambda)/P(\neg x_{*j}) \to 1
\end{aligned}
\tag{2.3}
$$

Therefore, the minimum ability of c^λ to predict y can approach that of x_{*j} by maximising its coverage $P(c^\lambda)$ and ability to predict x_{*j}: $P(x_{*j}|c^\lambda)$ (or $P(\neg x_{*j}|c^\lambda)$ if x_{*j}'s absence predicts y). In the example context of sentiment classification from bag-of-words features, a condition based on a word like "wonderful" that is common (relatively high $P(c^\lambda)$) helps predict the presence of positive words ($P(\text{"loved"}|\text{"wonderful"}) \gg P(\text{"loved"})$ where $P(y^+|\text{"loved"}) \gg P(y^+)$) and helps predict the absence of negative words ($P(\neg\text{"hated"}|\text{"wonderful"}) \gg P(\neg\text{"hated"})$ where $P(y^+|\neg\text{"hated"}) \gg P(y^+)$) also helps predict the positive class: $P(y^+|\text{"wonderful"}) \gg P(y^+)$. Because WITAN does not know which features predict classes through either their presence or absence, it should aim to minimise the conditional entropy of all features (maximising $\max(P(x_{*j}|c^\lambda), P(\neg x_{*j}|c^\lambda))\forall x_{*j}$). Similar entropy-based metrics have been used in hierarchical divisive clustering (HDC) to select features for splitting clusters: splits based on maximising the mean of the gain ratio (Qin, Ma, Herawan & Zain, 2014) or normalised information gain (Wei, Liang, Guo, Song & Sun, 2019) over all feature variables compared favourably to other criteria (Wei et al., 2019). While these algorithms

seek a single optimal clustering that assigns each instance to a single cluster, WITAN will select LFs that can partition instances in a variety of ways, allowing the user to choose which LFs best suit their classification task. This LF diversity is achieved by ensuring the full set of LFs reduces conditional entropy across many different features, e.g. "horror" predicts genre-aligned words that are not predicted by sentiment-aligned keywords.

In light of the discussion above, WITAN's utility function is based on the information gain achieved by LF λ on a binary feature x_{*j}:

$$IG(x_{*j}, c^{\lambda}) = H(x_{*j}) - H(x_{*j}|c^{\lambda}) \tag{2.4}$$

where the entropy of Boolean variable E is given by:

$$H(E) = -P(E)\log_2 P(E) - (1 - P(E))\log_2(1 - P(E)) \tag{2.5}$$

and the probabilities involved in the entropy calculations can be calculated from the training set:

$$P(x_{*j}) = \frac{1}{n}\sum_{i=1}^{n} x_{i,j} \qquad P(x_{*j}|c^{\lambda}) = \frac{\sum_{i=1}^{n} x_{i,j}c_i^{\lambda}}{\sum_{i=1}^{n} c_i^{\lambda}} \tag{2.6}$$

Because the coverage vectors are binary, there is no need to normalise the information gain or use a gain ratio to reduce bias towards multi-valued conditions (Wei et al., 2019). While some rule learners (e.g. PRISM, FOIL) formulate information gain with regards to the rule's target value for the class attribute (Fürnkranz & Flach, 2005), WITAN does not know whether the 0/1 value of a target feature will predict the class, so information gain must be formulated in terms of the entropy of both target feature values (similar to the decision tree definition of information gain).

The utility function U that WITAN will aim to maximise when selecting LFs can now be defined:

$$U_{X,\bar{H},\gamma,w}(\lambda) = \sum_{\substack{j=1,\\ j\notin d^{\lambda}}}^{m} w_j \sum_{\substack{i=1,\\ c_i^{\lambda}=1}}^{n} \max(0, \bar{h}_{i,j} - H(x_{*j}|c^{\lambda}))^{\gamma} \tag{2.7}$$

In essence, U sums the information gain achieved by LF λ across all features j over all covered instances i, with several parameters and characteristics of note:

- Values from an entropy matrix $\bar{H}$ are used in place of $H(x_{*j})$. $\bar{H}$ will initially mirror $H(x_{*j})$ (Algorithm 2.1, line 1), but its values can be iteratively updated by WITAN to account for the information gain of previously selected LFs.
- As an additional LF cannot detract from information already gained, negative gain is ignored through the use of $\max$.
- In order to reduce bias towards less interpretable LFs based on many features, information gain on any feature in d^{λ} is ignored.
- Because information gain is only summed over instances covered by c^{λ}, U favours higher coverage as desired.
- Initial experimentation found that LFs with high information gain over a smaller number of instances were more useful than those with weaker gain over many instances. Such LFs can be preferred by setting $\gamma > 1$, which increases the importance of high information gain. $\gamma = 2$ was found to be a reasonable setting in an ablation study (Section 2.5.5).
- Weights vector $w \in \{[0, +\infty)\}^m$ can be used to control the relative importance of gain on each feature. The use of w is discussed in Section 2.3.4, but for now, assume all weights are 1.

Minimising conditional entropy within the instances covered by an LF can have some counter-intuitive effects in practice, particularly when the prior probabilities of features

are low. For instance, in the previous example, if $0.5 > P(\text{“loved”}|\text{“wonderful”}) \gg P(\text{“loved”})$, then conditioning on “wonderful” will actually increase the conditional entropy of the feature “loved”. While this will not negatively impact the utility of “wonderful” (negative IG is ignored), it does mean sparse bag-of-words features are more likely to reduce in conditional entropy when they become even less frequent under the candidate condition. It is worth re-emphasising that minimising conditional entropy is still an appropriate strategy, given that it is unknown whether the presence or absence of a given feature will be predictive of a class value y.

2.3.2 Core of the Algorithm

The utility function U can now be used to formulate the core WITAN algorithm laid out in Algorithm 2.1. WITAN begins by initialising matrix $\bar{H}$ with the entropy of each feature for each instance in the training data X (line 1) and a set of candidate LFs $\hat{\Lambda}$ that each represent a simple condition on a single binary feature (line 2). The user may choose to constrain the search space of candidate LFs to include only those covering a minimum proportion $c^{\min} \in [0, 1]$ of training instances and may optionally specify a set of seed LFs $\tilde{\Lambda}$ to initialise the set of selected LFs Λ (lines 3–9).

Each iteration of WITAN’s main loop (lines 10–17) selects the candidate LF that maximises utility function U (line 11, Equation (2.7)) and adds it to the set of selected LFs Λ (line 13). After an LF λ is selected, the entropy matrix $\bar{H}$ used as the baseline entropy by U is updated (line 15). Entries for all instances i covered by the LF ($c_i^{\lambda} = 1$) are assigned the LF’s conditional entropies for each feature j ($H(x_{*j}|c^{\lambda})$) if those entropies are less than the current $\bar{H}_{i,j}$ entries (line 23). In this way, U’s information gain will always be computed against the minimum conditional entropy achieved by any selected LF matching a given instance. Because of this, each iteration selects a new LF that can provide information gain on instances not covered by other LFs

Algorithm 2.1: Core WITAN Algorithm.

Input : Training data X, seed LFs $\tilde{\Lambda}$, gain exponent γ, min. LF coverage $c^{\min}$, stopping coverage $C^{\min}$

Output : Set of proposed LFs Λ

$\bar{h}_{i,j} \leftarrow H(x_{*j})\ \forall\ i \in \{1,\dots,n\},\ j \in \{1,\dots,m\}$;
$\hat{\Lambda} \leftarrow \{\hat{\lambda}[d^{\hat{\lambda}} = \{j\};\ c^{\hat{\lambda}} = x_{*j}]\ \forall\ j \in \{1,\dots,m\} : \frac{1}{n}\sum_{i=1}^{n} c_i^{\hat{\lambda}} \geq c^{\min}\}$;
$\Lambda \leftarrow \{\}$;
foreach $\tilde{\lambda} \in \tilde{\Lambda}$ **do**
    $\Lambda \leftarrow \Lambda \cup \{\tilde{\lambda}\}$;
    $w \leftarrow$ `getWeights(`Λ`);`
    `updateEntropies(`$\bar{H}$, $\tilde{\lambda}$`);`
    `updateCandidates(`$\hat{\Lambda}$, $\tilde{\lambda}$`);`
end
while $\frac{1}{n}\sum_{i=1}^{n} \max(\{c_i^{\lambda}\ \forall\ \lambda \in \Lambda\}) < C^{\min}$ **do**
    $\lambda \leftarrow \arg\max_{\hat{\lambda} \in \hat{\Lambda}} U_{X,\bar{H},\gamma,w}(\hat{\lambda})$;
    $\lambda \leftarrow$ `extendLF(`λ`);`
    $\Lambda \leftarrow \Lambda \cup \{\lambda\}$;
    $w \leftarrow$ `getWeights(`Λ`);`
    `updateEntropies(`$\bar{H}$, λ`);`
    `updateCandidates(`$\hat{\Lambda}$, λ`);`
end
return Λ;

Procedure `getWeights(`Λ`):`
    $w_j \leftarrow 1\ \forall\ j \in \{1,\dots,m\}$;
    return w
Procedure `updateEntropies(`$\bar{H}$, λ`):`
    $\bar{h}_{i,j} \leftarrow \min(\bar{h}_{i,j}, H(x_{*j}|c^{\lambda}))\ \forall\ i,j : c_i^{\lambda} = 1$;
Procedure `updateCandidates(`$\hat{\Lambda}$, λ`):`
    $\hat{\Lambda} \leftarrow \hat{\Lambda} \setminus \{\lambda\}$;
Procedure `extendLF(`λ`):`
    return λ;

(increasing coverage) or on features not well predicted by other LFs (increasing LF diversity). Information gain would ideally be computed against the conditional entropy of the full set of selected LFs ($H(x_{*j}|c^{\lambda} \; \forall \; \lambda \in \Lambda)$), but the computational cost of fully re-computing $\bar{H}$ in each iteration would be prohibitive. Taking the minimum entropy implies a conservative assumption that the set of selected LFs has maximal information redundancy, i.e. only considering the amount of information gain by which a candidate LF exceeds the information gain of any previously selected LF assumes the worst-case maximal information redundancy between the candidate and previously selected LFs.

WITAN's main loop continues until the set of selected LFs Λ covers a user-specified proportion $C^{\min} \in [0, 1]$ of training instances (line 10), at which point Λ is returned so that the user may choose which LFs are relevant to their classification task and assign class labels to them.

WITAN's algorithm also contains several procedures for which subsequent subsections will provide alternative definitions to describe extensions of the core algorithm. These control the feature weights used by the utility function U (`getWeights`, which currently weights all features equally), how the set of candidate LFs $\hat{\Lambda}$ is updated after an LF is selected (`updateCandidates`, which currently just removes the selected LF from the candidate set), and how a selected candidate LF may be extended before being added to Λ (`extendLF`, which does not currently extend LFs).

2.3.3 Conjunctive and Disjunctive LFs

In the core WITAN algorithm, the best LF is selected from a set of candidates in each iteration, similar to IWS's approach of searching through a family of LFs (Boecking et al., 2021). To limit the computational cost in each iteration, both the core WITAN algorithm and IWS constrain the candidate set to LFs with conditions based on a single feature or heuristic. In this section, the core WITAN algorithm is extended to generate

LFs based on logical conjunctions (ANDs) and disjunctions (ORs) of feature conditions, incurring minimal additional computational cost by dynamically updating the candidate set and selected LFs. This is expected to improve the effectiveness of selected LFs, as conjunctive LFs can target narrower subsets of instances to achieve higher information gain, while disjunctive LFs can target wider sets of instances to achieve higher coverage.

Algorithm 2.2 redefines `updateCandidates` to extend the LF candidate set $\hat{\Lambda}$ to include logical conjunctions of a selected LF λ's condition with other feature conditions. Specifically, for each feature j that is not already part of the selected LF's condition ($j \notin d^{\lambda}$), a new conjunctive candidate LF $\hat{\lambda}$ is constructed such that $d^{\hat{\lambda}} = d^{\lambda} \cup \{j\}$ and $c^{\hat{\lambda}} = c^{\lambda} \wedge x_{*j}$, excluding LFs that do not achieve minimum coverage $c^{\min}$ (line 1). As the conjunction means that each new $\hat{\lambda}$ will only cover a subset of the instances already covered by λ, $\hat{\lambda}$ can be considered a *child* of λ. This forms a hierarchical structure of LFs, notated by defining the parent of $\hat{\lambda}$ as $p^{\hat{\lambda}} = \lambda$. This conjunctive structure enables the expression of LFs that identify subcategories of other LFs: an LF conditional on the word "movie" could be the parent of LFs conditional on words that identify how that movie is described, such as "horrible" or "liked". The new procedure definition also alters how selected LFs are removed from the candidate set to account for both conjunctive and disjunctive LFs: `updateCandidates` now removes any candidates that are siblings of the original LF λ and are conditioned by a subset or equal set of features (line 2). Finally, to prevent increasing computational cost through the candidate set's growth, the original size of m is maintained by only retaining candidates with the highest utility (line 3).

Algorithm 2.3 redefines `extendLF` to attempt extending a selected LF λ into a disjunction between the original LF's condition and other candidate LF conditions. Each iteration of `extendLF`'s main loop finds the sibling candidate LF $\hat{\lambda}'$ that maximises the utility function with weights w' that are relative to the information gain achieved by λ on each feature (lines 2–3). A sibling that provides information gain on a similar set

Algorithm 2.2: `updateCandidates` for conjunctive LFs.

Input : Training data X, gain exponent γ, min. LF coverage $c^{\min}$, entropy matrix $\bar{H}$, weights vector w, set of candidate LFs $\hat{\Lambda}$, newly selected LF λ

Output : Updated set of candidate LFs $\hat{\Lambda}$

1 $\hat{\Lambda} \leftarrow \hat{\Lambda} \cup \{\hat{\lambda}[d^{\hat{\lambda}} = (d^{\lambda} \cup \{j\});\ c^{\hat{\lambda}} = (c^{\lambda} \wedge x_{*j});\ p^{\hat{\lambda}} = \lambda]$
$\forall\ j \in \{1, \ldots, m\} : \frac{1}{n} \sum_{i=1}^{n} c_i^{\hat{\lambda}} \geq c^{\min},\ j \notin d^{\lambda}\}$;

2 $\hat{\Lambda} \leftarrow \hat{\Lambda} \setminus \{\hat{\lambda} \ \forall\ \hat{\lambda} \in \hat{\Lambda} : p^{\hat{\lambda}} = p^{\lambda},\ d^{\hat{\lambda}} \subseteq d^{\lambda}\}$;

3 $\hat{\Lambda} \leftarrow \underset{\hat{\Lambda}' \subseteq \hat{\Lambda},\ |\hat{\Lambda}'| \leq m}{\arg\max} \sum_{\hat{\lambda} \in \hat{\Lambda}'} U_{X,\bar{H},\gamma,w}(\hat{\lambda})$;

of features to λ tends to have a logical similarity with λ, which leads to a disjunctive condition that makes sense to the user. For example, an LF for the positive word "wonderful" could be extended with a disjunction to a similar positive word, such as "excellent". LFs λ and $\hat{\lambda}'$ are used to construct a new disjunctive LF λ' such that $d^{\lambda'} = d^{\lambda} \cup d^{\hat{\lambda}'}$ and $c^{\lambda'} = c^{\lambda} \vee c^{\hat{\lambda}'}$ (line 4). If λ' provides greater w'-weighted utility than the original LF λ, then it is considered a superior LF fulfilling the same role for feature diversity, and λ can be replaced by λ' (lines 5–6). Iteration continues to add further disjunctions to λ until there is no further increase in utility, at which point λ is returned so that it can be added to Λ (line 8).

Algorithm 2.3: `extendLF` for disjunctive LFs.

Input : Training data X, gain exponent γ, entropy matrix $\bar{H}$, set of candidate LFs $\hat{\Lambda}$, newly selected LF λ

Output : Extended selected LF λ

1 **loop**
2 $\quad w'_j \leftarrow \max(0, IG(x_{*j}, c^{\lambda}))\ \forall\ j \in \{1, \ldots, m\}$;
3 $\quad \hat{\lambda}' \leftarrow \underset{\hat{\lambda} \in \hat{\Lambda},\ p^{\hat{\lambda}} = p^{\lambda}}{\arg\max} U_{X,\bar{H},\gamma,w'}(\hat{\lambda})$;
4 $\quad \lambda' \leftarrow \lambda'[d^{\lambda'} = (d^{\lambda} \cup d^{\hat{\lambda}'}); c^{\lambda'} = (c^{\lambda} \vee c^{\hat{\lambda}'}); p^{\lambda'} = p^{\lambda}]$;
5 $\quad$ **if** $U_{X,\bar{H},\gamma,w'}(\lambda') > U_{X,\bar{H},\gamma,w'}(\lambda)$ **then**
6 $\quad\quad \lambda \leftarrow \lambda'$;
7 $\quad$ **else**
8 $\quad\quad$ **return** λ;
9 $\quad$ **end**
10 **end**

2.3.4 Incorporating Feedback

Discussion of WITAN has so far highlighted the benefits of being able to apply it in an unsupervised fashion, namely that the user does not need to have a fixed idea of their class structure before WITAN can begin LF generation; input is only required from the user to assign class labels to WITAN-generated LFs. However, other LF generation methods like IWS and Snuba use some form of user supervision to guide the generation of LFs suited to the user's classification task. This section describes an approach for guiding WITAN's selection of LFs with feedback on seed and previously generated LFs.

The user can provide feedback by marking any LF in the set of selected LFs Λ as `approved`. Feedback on an LF λ is represented with $f^{\lambda} \in \{\texttt{approved}, \texttt{null}\}$, where $f^{\lambda} = \texttt{null}$ when the user has not provided feedback for λ. Given feedback for a subset of selected LFs, WITAN should select subsequent LFs that support a class structure consistent with approved LFs: $\{\lambda \ \forall \ \lambda \in \Lambda : f^{\lambda} = \texttt{approved}\}$. One way of interpreting this feedback is that `approved` LFs have successfully achieved information gain on features that are correlated with the target class variable. For example, suppose the user approves an LF for the positive word "wonderful" that provides information gain on other positive and negative words. In that situation, WITAN should favour LFs that also provide information gain on such words, as they will likely also be effective classifiers of positive/negative sentiment. Therefore, WITAN can utilise feedback by using it to update the weight w_j of information gain for each feature j in utility function U, increasing the weight of features for which `approved` LFs provided high information gain. The intent is to prioritise LFs for classes complementary to `approved` LFs, as such LFs will optimise WITAN's utility function by further reducing conditional entropy on similar feature sets while covering different sets of instances.

Algorithm 2.4 redefines `getWeights` to incorporate feedback f^{λ} for selected LFs $\lambda \in \Lambda$. If no feedback is available, equal weights are returned (line 3). Otherwise,

Algorithm 2.4: `getWeights` for incorporating feedback.

Input : Training data X, selected LFs Λ
Output : Weights vector w

1 $\Lambda^f \leftarrow \{\lambda \in \Lambda : f^\lambda = \text{approved}\}$;
2 **if** $\Lambda^f = \varnothing$ **then**
3 $\quad w_j \leftarrow 1 \; \forall \; j \in \{1, \ldots, m\}$;
4 **else**
5 $\quad w_j \leftarrow 0 \; \forall \; j \in \{1, \ldots, m\}$;
6 $\quad$ **foreach** $\lambda \in \Lambda^f$ **do**
7 $\quad\quad w'_j \leftarrow \max(0, IG(x_{*j}, c^\lambda)) \; \forall \; j$;
8 $\quad\quad w_j \leftarrow w_j + \frac{w'_j}{\sum_{l=1}^{m} w'_l} \; \forall \; j$;
9 $\quad$ **end**
10 **end**
11 **return** w

weights are computed as follows. For each `approved` LF λ, Vector w' is constructed of the information gains achieved by λ on each feature (line 7). The final weights vector w is constructed by summing normalised w' for all `approved` LFs (line 8), such that the sum of weights contributed by each LF equals 1. Note that in Algorithm 2.1, the weights are updated in each iteration (line 14), allowing the user to interactively guide WITAN by immediately providing feedback on each LF selected by WITAN.

2.3.5 Analysis of WITAN

The following section analyses WITAN to highlight the interaction modes it affords, assess its computational complexity, qualitatively analyse WITAN-generated LFs, and discuss potential failure modes.

Interaction Modes

Because WITAN can be run without any labelled instances, seed LFs, or even predefined class structure, it achieves the primary goal of performing LF generation without any initial supervision. Once a set of LFs has been generated, the user can choose which LFs

to assign class labels to. This can be considered a straightforward task for the user, as the conjunctive and disjunctive feature-based conditions generated by WITAN are simply expressed in conjunctive normal form (an "AND of ORs"). Similar decision rules have been found to be a highly understandable format for expressing the behaviour of machine learning models to users (Stumpf et al., 2007). Additionally, a user study with IWS's single-condition LFs found that users could not only label LFs accurately, but they could also label them faster than they could label individual instances (Boecking et al., 2021).

Users may also provide initial seed LFs for WITAN to grow with additional LFs that increase coverage and feature diversity. Furthermore, once a user has labelled a set of WITAN-generated LFs, they may use them as seed LFs in a subsequent execution of WITAN. With the feedback extension described in Section 2.3.4, WITAN can use the knowledge of whether or not a generated LF was approved and assigned a class label to guide LF generation towards those that are more likely to be useful for the user's target classification task.

In the extreme case, a user may operate WITAN in a step-by-step manner: generating a single LF, labelling that LF, and then continuing to run WITAN with feedback. Such a step-by-step approach is not suitable when the user must review a full set of LFs to determine the class structure, but rather when the user already knows their desired class structure (as required for feedback-based LF generators like IWS) or would be able to recognise LF conditions that identify classes relevant to their classification task. For example, a user starting to classify warranty claims according to fault type may not know what types of fault exist but would recognise that LF conditions on keywords like "battery" or "charging" clearly identify a battery-related fault type. Experiments are performed with step-by-step labelling in Section 2.5.5 to demonstrate the feedback extension.

Computational Complexity

The computational complexity of each WITAN iteration is dominated by the computation of utility function U for each candidate LF $\hat{\lambda} \in \hat{\Lambda}$. While the $O(n)$ computation of $H(x_{*j}|c^{\hat{\lambda}})$ for each j and $\hat{\lambda}$ can be cached between iterations, performing the sum over m features and n instances in U for each of up to m candidate LFs still results in a complexity of $O(m^2 n)$. Therefore, the runtime of WITAN is largely determined by the number of features, as demonstrated in the runtime study (Section 2.5.3).

The computational complexity of WITAN is not affected by the proposed extensions to the core algorithm. `updateCandidates` in Algorithm 2.2 computes U for up to m new candidates for a complexity of $O(m^2 n)$, and also ensures the number of candidates does not exceed m. `extendLF` in Algorithm 2.3 can use cached candidate conditional entropies and re-weight them by w', leaving the $O(mn)$ computation of $U_{X,\bar{H},\gamma,w'}(\lambda')$ in each of a maximum of m loop iterations as the dominating factor. The complexity of computing the Boolean combination of features to produce a coverage vector c^{λ} for an LF added by these extensions is at most $O(mn)$, so it does not impact the overall complexity. Finally, `getWeights` in Algorithm 2.4 performs the $O(n)$ computation of IG for m features and each $\lambda \in \Lambda^f$, giving complexity $O(|\Lambda^f|mn)$, where $|\Lambda^f|$ is expected to be much smaller than m and the entropies used to compute IG could be cached.

WITAN's memory requirements are dominated by entropy matrix $\bar{H}$ and candidate coverage vectors $c^{\hat{\lambda}} \; \forall \; \hat{\lambda} \in \hat{\Lambda}$ that both require $O(mn)$ space, as well as cached candidate conditional entropies $H(x_{*j}|c^{\hat{\lambda}}) \; \forall \; j \in \{1, \ldots, m\}, \hat{\lambda} \in \hat{\Lambda}$ that require $O(m^2)$ space.

Qualitative Analysis

Fig. 2.2 presents LFs generated by ten iterations of WITAN on bag-of-words features from three text classification datasets, colouring LFs that were accurate enough for a

IMDb Review Sentiment

- **negative:** waste∨worst
  (coverage: 13%, accuracy: 90%)
- documentary
- movie
  - **negative:** ∧ waste∨bad∨stupid∨crap
    (coverage: 22%, accuracy: 79%)
- **positive:** wonderful∨excellent∨superb
  (coverage: 14%, accuracy: 80%)
- films
- animation
- episodes∨episode
- **positive:** loved
  (coverage: 5%, accuracy: 75%)
- rent

Bias Bios: Painter or Architect

- **painter:** paintings∨painting
  (coverage: 25%, accuracy: 99%)
- **architect:** architecture∨development∨architect∨software∨architectural∨solutions∨management
  (coverage: 34%, accuracy: 95%)
- **painter:** art∨gallery∨artist (coverage: 30%, accuracy: 88%)
- **architect:** data∨enterprise (coverage: 5%, accuracy: 98%)
- **architect:** experience∨projects
  (coverage: 19%, accuracy: 86%)
- **painter:** paints∨life∨paint∨color
  (coverage: 15%, accuracy: 89%)
- **architect:** microsoft∨technologies∨business∨services∨applications∨web∨application
  (coverage: 13%, accuracy: 91%)
- **painter:** york (coverage: 8%, accuracy: 71%)
- **painter:** colors (coverage: 2%, accuracy: 97%)
- **painter:** images∨work∨arts∨born∨school∨landscape
  (coverage: 43%, accuracy: 70%)

20Newsgroups Topics

- nntp
- **computer:** thanks
  (coverage: 15%, accuracy: 57%)
  - **computer:** ∧ advance
    (coverage: 3%, accuracy: 68%)
- article
  - **religion:** ∧ rutgers
    (coverage: 3%, accuracy: 87%)
- **sports:** bike
  (coverage: 2%, accuracy: 100%)
- **computer:** hi∨windows∨pc∨graphics∨dos∨mac (coverage: 18%, accuracy: 80%)
- **sports:** hockey∨season∨baseball
  (coverage: 6%, accuracy: 94%)
- **science:** clipper
  (coverage: 3%, accuracy: 97%)
- **religion:** god∨christians∨christian∨jesus∨bible∨christianity∨religion∨waco∨jews∨christ∨religious∨church∨israel
  (coverage: 18%, accuracy: 56%)

Figure 2.2: Examples of WITAN-generated LF sets.

simulated user to assign a class label (as described in Section 2.5). Unlabelled LFs are shown in grey. "$\vee$"-separated keywords form disjunctive (OR) LF conditions and nested LFs prefixed by "$\wedge$" represent conjunctions (ANDs) between parent and child LFs.

Overall, the LFs tend to use coherent sets of keywords with understandable meanings while also achieving desirable levels of coverage and accuracy. Note that while positive/negative sentiment analysis is the classification target for the IMDb reviews, unsupervised WITAN produces a diverse range of LFs that could be used to classify reviews based on other concepts (e.g. by format: "movie" vs "episode"; or genre: "documentary" vs "animation"). In the dataset for discriminating painter and architect biographies, WITAN even begins to identify LFs that could be associated with sub-classes of the intended target classes, such as "data/enterprise" architects. Finally, note that WITAN is able to capture a range of classes in the multi-class 20Newsgroups dataset, though only one LF is identified for the rarer "science" class, and no LFs are identified for the rare "politics" class after only ten iterations.

Potential Failure Modes

Finally, this section discusses scenarios where WITAN's approach to generating LFs may produce a sub-optimal classifier. Recall that the intuition behind WITAN's utility function presented in Section 2.3.1 is based on the assumption that some features will be correlated with target classes. Therefore, WITAN is unlikely to produce accurate LFs if no features correlate with the target classes. Furthermore, as WITAN favours LFs with high coverage, it may struggle to identify LFs for rare classes, such as in classification tasks with high imbalance or numerous target classes (as demonstrated with the 20Newsgroups LFs in Section 2.3.5 and the multi-class experiments of Section 2.5.4). This issue may be addressed in some cases by increasing γ to prioritise fine-grained LFs with higher information gain over those with higher coverage or by providing feedback to focus WITAN on LFs that are complementary to selected LFs. Accurately

representing rare classes is a general problem for weak supervision from coarse LFs, and future work could potentially augment WITAN with instance-level labelling.

2.4 Multi-class Extension of IWS

The experimental study in Section 2.5 seeks to go beyond the almost solely binary-class experiments of prior studies on data programming by performing experiments with multi-class datasets. Including the IWS LF generation method in these experiments requires adjustments to the original algorithm. While the authors of IWS note that their ideas extend to the multi-class case, their described algorithm is limited to binary classification (Boecking et al., 2021). IWS uses feedback on proposed LFs to train an ensemble of neural networks that identify candidate LFs the user is likely to approve with positive feedback. This requires a vector representation $\tau_0(\lambda)$ for a given LF λ, which they achieve by using the output of each LF on the training set: $\tau_0(\lambda) = [\lambda(x_1), \ldots, \lambda(x_n)]$ assuming $\lambda(x) \in \{-1, 0, 1\}$ where -1 and 1 represent the binary classes and 0 is returned when abstaining. $\tau_0(\lambda)$ is then projected to dimensionality $d' < n$ via PCA to produce the final representation for the neural networks. Because the neutral abstention value between the positive/negative class split is tailored to binary classification, this section proposes two alternative representations for $\tau_0(\lambda)$ that can support an arbitrary number of class values and still be projected to a reduced dimensionality.

The first proposed variant of IWS, *IWS-Multi*, constructs $\tau_0^{\text{multi}}(\lambda)$ with an entry for each instance x and possible class value y. Each entry equals 0 when λ abstains

and equals 1 or -1 when the class value returned by $\lambda(x)$ does or does not equal y respectively:

$$\tau_0^{\text{multi}}(\lambda) = [\mu(\lambda, x_1, y_1), \ldots, \mu(\lambda, x_n, y_1), \ldots, \mu(\lambda, x_1, y_k), \ldots, \mu(\lambda, x_n, y_k)]$$

$$\text{where:} \quad \mu(\lambda, x, y) = \begin{cases} 0 & \text{if } \lambda(x) = \varnothing \\ 1 & \text{if } \lambda(x) = y \\ -1 & \text{otherwise} \end{cases} \tag{2.8}$$

This representation should better retain the meaning behind abstaining than simply extending the original paper's representation with additional integers to represent additional classes.

The second proposed variant of IWS, *IWS-Distinct*, constructs $\tau_0^{\text{distinct}}(\lambda)$ with a single Boolean entry for each instance x indicating whether λ assigns any class or abstains:

$$\tau_0^{\text{distinct}}(\lambda) = [\delta(\lambda, x_1), \ldots, \delta(\lambda, x_n)]$$

$$\text{where:} \quad \delta(\lambda, x) = \begin{cases} 0 & \text{if } \lambda(x) = \varnothing \\ 1 & \text{otherwise} \end{cases} \tag{2.9}$$

The reasoning behind this representation is that a separate LF candidate should not be needed for each possible target class, as the user can be asked to assign the class label in the same way required by WITAN. The intuition is that if a user can be expected to assess whether or not an LF condition is accurate for a given class, they should easily be able to determine the most appropriate class for an LF condition. In this setting, $\tau_0(\lambda)$ only needs to represent which instances the LF covers, which is captured by $\tau_0^{\text{distinct}}(\lambda)$.

2.5 Experimental Study

The following experiments compare WITAN to state-of-the-art LF generation methods IWS (Boecking et al., 2021) and Snuba (Varma & Ré, 2018), LFs produced by clustering methods HDC (Wei et al., 2019) and CBI (Forman et al., 2015), the common low-effort labelling approaches semi-supervised learning (Zhou, Bousquet, Lal, Weston & Schölkopf, 2004) and active learning (Lewis & Gale, 1994), as well as a baseline of labelling a random subset of training instances (*random labelling*). These experiments demonstrate that, even without initial supervision, WITAN can achieve competitive F1 scores on both binary and multi-class classification tasks. Method runtimes are also compared, and an ablation study is performed to justify WITAN's design decisions. All experiments were performed on 64-bit Ubuntu 20.04 running on a 12x2.60GHz Intel Core i7 CPU with 48GB of memory. Source-code for the experiments (including dataset pre-processing) has been made available online at: `https://github.com/ben-denham/witan`.

2.5.1 Experiment Framework

As the downstream goal of labelling is to produce an effective classifier, performance is evaluated for a classifier trained using each labelling method. In each experiment, a given labelling method is applied to an unlabelled training dataset to produce labels for all or a subset of instances. For methods that produce LFs, probabilistic training labels are generated from the LFs using the Snorkel labelling model (A. Ratner et al., 2017) under assumptions of independent LFs and balanced classes[1]. Labelled training instances are then embedded into 300 feature dimensions via truncated Singular Value Decomposition (SVD) and used to train a multilayer perceptron with two hidden

[1] Although this assumption will not hold for all datasets, it is a realistic operating condition when the user does not know the true class balance.

layers of 20 units, RELU activations, softmax output, and logarithmic loss. This is the same discriminative model used to evaluate IWS (Boecking et al., 2021), except the sigmoid output has been replaced with softmax to support multi-class classification. Classification performance is evaluated by the mean F1 score achieved across target classes on the ground-truth labels of a held-out test set. While the AUC metric used in the evaluation of IWS (Boecking et al., 2021) evaluates performance across a range of classification thresholds, it is important to focus on the performance of the default threshold in the context of low-effort labelling, given that a user will likely not have access to labelled instances with which to tune the threshold. F1 also has a precedent in past evaluations of data programming (A. J. Ratner et al., 2016; Varma & Ré, 2018; Alonso Doval, 2021).

In order to compare labelling methods under similar conditions of user effort, each experiment specifies a budget of "user interactions" (Interaction Count IC) that a method may leverage. The form of each interaction varies depending on the nature of the method, but the user effort required for each interaction is approximately equivalent. For methods that accept input in the form of instance labels (Snuba, semi-supervised learning, active learning, and random labelling), each provided label is considered a single interaction. For LF-based methods (WITAN, IWS, HDC, and CBI), each LF the user is asked to approve or reject is counted as an interaction. For IWS-Distinct, WITAN, and CBI, each interaction also includes the selection of the class label that an approved LF should assign. If the interaction budget is less than the number of candidate LFs, surplus interactions will effectively be unused. These user interactions are simulated in experiments based on ground truth labels for training instances: instance-based methods may query specific instance labels while LFs are approved if they would be at least 20% more accurate on the training dataset than an ideal random classifier for the target class (acc $\geq \frac{1}{k} + 0.2$, assuming balanced classes in the absence of knowledge of the class distribution). As WITAN, IWS, and CBI can also accept seed LFs, seeded

experiments are performed with two seed LFs for each class based on randomly selected features with accuracy: $0.2 \leq (\text{acc} - \frac{1}{k}) \leq 0.35$. The described framework of counting interactions (specifically, equating instance labelling and LF approval), simulating users, and providing seed LFs is based on that used to evaluate IWS (Boecking et al., 2021).

Except for IWS requiring the full set of classes for initialisation, the user simulated in this experimental framework is not expected to have prior knowledge of the class structure, but rather enough understanding of their classification task to identify the correct class for a given instance or LF condition. While extensive understanding would be required to identify instance classes for instance-based methods or to identify classes for HDC's LF conditions (where the conditioned features increase with the number of LFs), identifying classes for the LF conditions of WITAN and CBI is expected to require minimal prior knowledge, as discussed in Section 2.3.5. This claim is justified by arguing that if an LF condition is relevant enough to a proposed class for the user to approve an IWS-generated LF, then they should just as easily be able to identify the correct class for such an LF condition. This also justifies including LF class label selection in each user interaction for WITAN and CBI. Also note that, under this experimental framework, any of the evaluated methods may fail to label instances for one or more classes after a budgeted number of interactions, in which case the evaluated classifier will not be trained on those classes.

Mean F1 scores are presented for five random seedings of methods and seed LFs, except for unseeded WITAN and HDC, which do not depend on a random seed. Standard deviations are omitted for brevity but never exceed 0.3. The median standard deviation for each method across experiments in each of Table 2.3, Fig. 2.4, and Fig. 2.5 never exceeds 0.090, except for semi-supervised learning, which never exceeds 0.135. Because runtimes are reported for non-parallelised experiments that are more time-consuming to execute, they represent a single random seeding.

Evaluated Methods

WITAN is evaluated with the extensions for conjunctive and disjunctive LFs described in Section 2.3.3 and without them (referred to as WITAN-Core). Instead of stopping according to $C^{\min}$, WITAN generates a number of LFs equal to the allowed IC, and each LF is either rejected or labelled by the simulated user. While both WITAN variants include the feedback extension (Section 2.3.4), only seed LFs are marked as `approved`. Without seed LFs, both variants test WITAN's ability to generate an entire set of LFs without supervision, while interactive user feedback is evaluated in a subsequent ablation study (Section 2.5.5). Except as noted in the ablation study, WITAN is run with $\gamma = 2$ and $c^{\min} = 0.02$.

WITAN is compared to two state-of-the-art LF generation methods: IWS (Boecking et al., 2021) and Snuba (Varma & Ré, 2018). The implementations used in these experiments are based on each paper's accompanying source-code. For IWS, experiments are performed with the AS and LSE-AC ($\tilde{m} = 100$) LF acquisition strategies. IWS's original LF representation of τ_0 is used for binary classification, and the IWS-Multi and IWS-Distinct variants proposed in Section 2.4 are used for multi-class classification. For Snuba, some binary-class assumptions in the original implementation are generalised to support multi-class classification. Additionally, decision stumps are used as heuristic models to generate reasonably interpretable LFs, and the number of instances randomly sampled for the input dataset from the training set is equal to IC. The formulation of ν in the paper is used instead of that in the codebase, as $m + 1$ in the divisor ensures ν is never negative. However, some implementation details of the codebase are retained: the range of $[0.25, 0.45]$ for β (which the codebase states achieves better results) and keeping three LFs from the first Snuba iteration. The fixed number of 20 iterations implemented in the codebase is also used rather than the stopping condition described in the paper. The fixed number of iterations is used because hyperparameter values were

not specified in the paper for the stopping condition and because initial experimentation with the stopping condition was less promising than with fixed iterations.

The experiments also evaluate using clustering to generate LFs that can be subsequently labelled by users. Hierarchical divisive clustering (HDC) is evaluated by implementing and applying the foundational MGR algorithm (Qin et al., 2014) with improvements to select conditions based on the mean normalised information gain (MNIG) and to always bisect the largest sub-cluster (Wei et al., 2019). Unlike most clustering methods, HDC's clusters are described by feature conditions, though its iterative bisections result in complex conditions with up to as many features as there are clusters. Clustering by intent (CBI; Forman et al., 2015) is also evaluated. CBI produces clusters with interpretable conditions but relies on initial seeding and interactive user feedback. The implementation produced for these experiments trains the "residual" classifier on instances covered without disagreement by seed LFs and uses the same model as the downstream classifier. Hyperparameter values from the original paper are used, except up to 1000 instances are always taken for the residual, capped at 50% of the training set for smaller datasets.

Finally, the experiments evaluate semi-supervised learning and active learning, which are commonly used approaches for learning with few labels, along with a baseline of random instance labelling. For semi-supervised learning, label spreading (Zhou et al., 2004) is applied with a KNN kernel to a random sample of training instances equal to the allowed interaction count. Active learning is initialised with random training instances until instances from at least two classes have been selected. Uncertainty sampling (Lewis & Gale, 1994) is then applied iteratively to query for additional training labels until the interaction budget is exhausted. In each uncertainty sampling iteration, the downstream classification model described above is trained with the current set of labelled training instances, and the training instance classified with the least certainty is queried for its label.

Datasets

Table 2.2 lists the binary-class and multi-class text classification datasets used in the experiments. Text is transformed into word count feature representations using the pipeline from the IWS codebase (Boecking et al., 2021). Binary bag-of-words features with at least 2% coverage of training instances are used as candidate LF conditions in IWS, Snuba, and WITAN (these feature counts are shown in parentheses next to the raw feature count in Table 2.2). The word `reuters` and other boilerplate text identifying the non-fake news source were removed from the FNS dataset. Many of these datasets have been used in other studies on weak supervision and data programming (Varma & Ré, 2018; Wang et al., 2021; Boecking et al., 2021), and some feature in the recently proposed WRENCH benchmark datasets for weak supervision (J. Zhang et al., 2021). Predefined train/test splits are used where available for consistency with other studies; otherwise, a random 50/50 split is used. All datasets except for ATW, SPM, TWN, and NYT have approximately balanced class distributions. DMG and SPM were retrieved from the UCI Machine Learning Repository (Dua & Graff, 2017). IMG is the subset of IMD reviews for titles belonging to either, but not both, of the most common genres (Drama or Comedy); successfully performing different classification tasks on the same input data highlights WITAN's ability to produce diverse LFs without initial supervision. Text datasets are the target of these experiments because they have been the primary focus of past data programming research and because bag-of-words features are a natural fit for the creation of LFs that can be easily interpreted by users.

2.5.2 Binary-Class Experiments

Table 2.3 presents the performance of both unseeded and seeded labelling methods on binary classification tasks, including the performance achieved by a classifier trained on the fully labelled training set. The F1 scores achieved after 25 and 100 user interactions

Table 2.2: Binary and multi-class datasets used in experiments.

	Description	**Train** n	k	m **(cov.** $\geq 2\%$**)**
IMD	IMDb review sentiment (Maas et al., 2011)	25,000	2	18,525 (791)
IMG	IMDb review drama/comedy	15,261	2	13.761 (770)
BPA	Painter/architect bio (De-Arteaga et al., 2019)	6,118	2	3,130 (246)
BPT	Professor/teacher bio (De-Arteaga et al., 2019)	12,294	2	4,858 (236)
BJP	Journalist/photographer bio (De-Arteaga et al., 2019)	16,129	2	5,597 (217)
BPP	Professor/physician bio (De-Arteaga et al., 2019)	27,238	2	6,752 (222)
AZN	Amazon review sentiment (He & McAuley, 2016)	160,000	2	22,130 (160)
YLP	Yelp review sentiment (X. Zhang, Zhao & LeCun, 2015)	19,000	2	7,868 (474)
PLT	Plots: action/romance (Varma & Ré, 2018)	973	2	211 (61)
FNS	Fake news identification (Ahmed, Traore & Saad, 2018)	22,449	2	21,179 (1,626)
BDB	DBPedia: politics/company (X. Zhang et al., 2015)	40,022	2	9,161 (178)
BAG	AGNews: business/tech (X. Zhang et al., 2015)	29,979	2	7,454 (137)
ATW	Airline tweet sentiment (Crowdflower, 2019)	5,771	2	846 (60)
DMG	Identifying disaster tweets (Mouzannar et al., 2018)	2,915	2	1,143 (91)
SPM	SMS spam identification (Almeida, Hidalgo & Yamakami, 2011)	2,786	2	439 (42)
TWN	20Newsgroups topics (Lang, 1995)	8,935	5	11,948 (799)
DBP	DBPedia categories (X. Zhang et al., 2015)	56,000	14	12,682 (142)
AGN	AGNews topics (X. Zhang et al., 2015)	60,000	4	12,494 (101)
NYT	NYT topics (Meng, Shen, Zhang & Han, 2018)	15,999	9	27,029 (2,372)

Table 2.3: Binary classification F1 scores for unseeded and seeded labelling methods at 25 and 100 interactions (IC).

Method	**IC**	**IMD**	**IMG**	**BPA**	**BPT**	**BJP**	**BPP**	**AZN**	**YLP**	**PLT**	**FNS**	**BDB**	**BAG**	**ATW**	**DMG**	**SPM**
Full supervision		0.836	0.780	0.950	0.898	0.933	0.942	0.905	0.872	0.779	0.976	0.995	0.901	0.822	0.964	0.941
WITAN	25	0.727	**0.710**	**0.923**	**0.848**	**0.892**	0.860	**0.772**	**0.737**	**0.675**	**0.900**	**0.979**	**0.777**	0.510	0.723	**0.816**
	100	0.753	0.768	0.931	0.817	0.887	0.875	0.779	0.795	0.624	0.905	0.949	0.708	0.550	0.690	0.781
WITAN-Core	25	**0.738**	0.703	0.910	0.836	0.859	0.840	0.760	0.737	0.667	0.891	0.978	0.728	**0.632**	0.783	0.745
	100	**0.785**	**0.771**	**0.938**	0.833	**0.900**	0.848	**0.834**	**0.801**	0.719	**0.908**	**0.976**	0.764	0.594	0.846	0.843
IWS-AS	25	0.363	0.511	0.619	0.398	0.504	0.660	0.559	0.427	0.635	0.448	0.774	0.479	0.536	0.676	0.536
	100	0.575	0.746	0.815	0.477	0.845	0.835	0.796	0.565	**0.721**	0.391	0.965	0.688	0.607	0.808	0.805
IWS-LSE-AC	25	0.000	0.354	0.793	0.524	0.640	0.839	0.574	0.398	0.467	0.375	0.938	0.525	0.526	0.634	0.675
	100	0.656	0.594	0.850	0.509	0.826	0.850	0.790	0.651	**0.721**	0.476	0.964	0.668	0.604	0.809	0.805
Snuba	25	0.458	0.436	0.718	0.726	0.693	0.820	0.507	0.497	0.601	0.584	0.781	0.489	0.519	0.781	0.624
	100	0.501	0.521	0.844	0.763	0.759	0.830	0.651	0.668	0.631	0.796	0.862	0.647	0.503	0.783	0.598
HDC	25	0.000	0.388	0.663	0.746	0.688	0.802	0.576	0.495	0.596	0.788	0.753	0.749	0.403	**0.865**	0.777
	100	0.545	0.428	0.686	0.746	0.695	0.862	0.572	0.500	0.714	0.836	0.759	**0.806**	0.442	0.894	0.452
Semi-supervised	25	0.543	0.464	0.664	0.609	0.743	0.450	0.521	0.468	0.536	0.796	0.861	0.436	0.450	0.502	0.575
	100	0.607	0.599	0.551	0.567	0.491	0.590	0.700	0.467	0.558	0.642	0.913	0.592	0.546	0.789	0.571
Active learning	25	0.551	0.562	0.846	0.739	0.739	0.860	0.622	0.596	0.583	0.799	0.894	0.519	0.568	0.761	0.728
	100	0.666	0.642	0.917	**0.837**	0.845	**0.911**	0.787	0.745	0.684	0.895	0.973	0.780	**0.715**	**0.914**	**0.876**
Random labelling	25	0.504	0.530	0.849	0.768	0.730	**0.864**	0.608	0.568	0.609	0.788	0.893	0.602	0.601	0.831	0.603
	100	0.683	0.590	0.900	0.820	0.835	0.907	0.776	0.730	0.672	0.892	0.964	0.765	0.695	0.891	0.705
Seeded WITAN	25	**0.774**	**0.761**	**0.931**	0.834	**0.890**	**0.880**	0.742	**0.795**	**0.696**	**0.909**	**0.980**	**0.768**	0.544	0.759	0.696
	100	0.779	0.748	0.931	0.806	0.880	**0.881**	0.808	**0.802**	0.616	0.903	0.960	0.726	0.570	0.573	0.696
Seeded WITAN-Core	25	0.756	0.756	0.926	**0.840**	0.863	0.838	**0.785**	0.768	0.680	0.893	0.974	0.756	**0.654**	**0.792**	**0.767**
	100	**0.790**	**0.763**	**0.941**	**0.822**	**0.900**	0.861	**0.835**	0.796	0.718	**0.915**	**0.976**	**0.770**	0.595	**0.846**	**0.843**
Seeded IWS-AS	25	0.703	0.657	0.752	0.576	0.644	0.763	0.646	0.594	0.678	0.649	0.899	0.600	0.645	0.716	0.637
	100	0.771	0.721	0.912	0.606	0.849	0.836	0.810	0.736	**0.721**	0.522	0.968	0.687	0.607	0.805	0.806
Seeded IWS-LSE-AC	25	0.333	0.393	0.877	0.652	0.705	0.842	0.632	0.443	0.659	0.521	0.942	0.610	0.621	0.761	0.725
	100	0.613	0.457	0.904	0.602	0.834	0.849	0.784	0.721	**0.721**	0.750	0.967	0.686	**0.607**	0.821	0.806
Seeded CBI	25	0.554	0.596	0.708	0.672	0.619	0.715	0.510	0.594	0.525	0.740	0.722	0.593	0.428	0.497	0.620
	100	0.617	0.603	0.796	0.744	0.781	0.778	0.507	0.653	0.525	0.809	0.926	0.620	0.428	0.497	0.620

(IC) are presented, representing low and moderate user effort, respectively. The best-performing methods for each dataset after each IC are highlighted in bold, revealing the following insights:

- For $IC = 25$, WITAN is the best-performing method. This low interaction setting is important not only because it requires less user effort, but also because a smaller set of LFs will be more understandable and maintainable.
- For $IC = 100$, WITAN-Core is the best-performing method. For all but three datasets, WITAN-Core achieves F1 within 0.1 of the fully supervised baseline, highlighting the ability of WITAN to train an accurate classifier with moderate labelling effort. Performance differences between WITAN and WITAN-Core are discussed in the ablation study (Section 2.5.5).
- Active learning is the next best performer, particularly with more interactions. Active learning's superior performance on certain classification tasks suggests that these tasks are better learned with specific example instances than with coarse LFs that cover many instances. However, it is worth emphasising that LFs have the additional benefit over labelled instances of acting as an understandable and maintainable knowledge representation.
- As expected, seeded methods tend to perform better than their unseeded counterparts. In particular, IWS without seeding sometimes fails to generate any approved rules (F1 = 0) for $IC = 25$. However, this difference is often small, particularly after more interactions.

Methods were compared across datasets using Friedman tests and accompanying Nemenyi post hoc tests (Demšar, 2006) for $IC = 25$ and $IC = 100$, and in both cases the null hypothesis that "all methods produce equivalent F1 scores" was rejected at the 95% confidence level. The critical difference diagrams in Fig. 2.3 show groups of methods with statistically insignificant differences. Note that all WITAN variants and

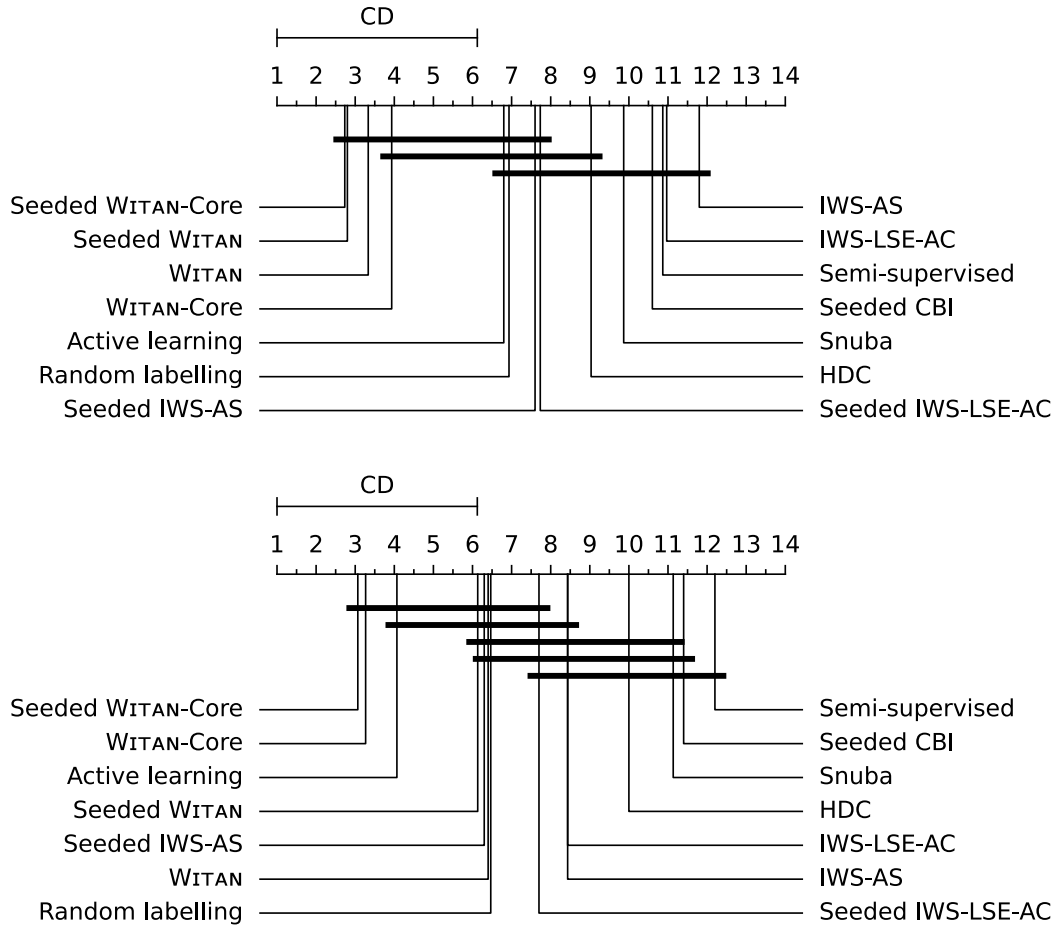


Figure 2.3: Critical difference diagrams for binary classification F1 of labelling methods at IC of 25 (above) and 100 (below).

active learning are in the top group for both values of IC and that seeded variants of WITAN achieve the best overall performance.

Fig. 2.4 further demonstrates how performance changes with IC. Note that the performance of most methods tends to converge after many interactions, but WITAN achieves near-optimal performance with far fewer interactions than other methods. It should also be noted that performance may drop at higher IC for LF-based methods if additional labelled LFs are of poorer quality or result in an incorrect class balance in instance labels; poor performance on even one class due to incorrect balance can greatly affect F1 scores.

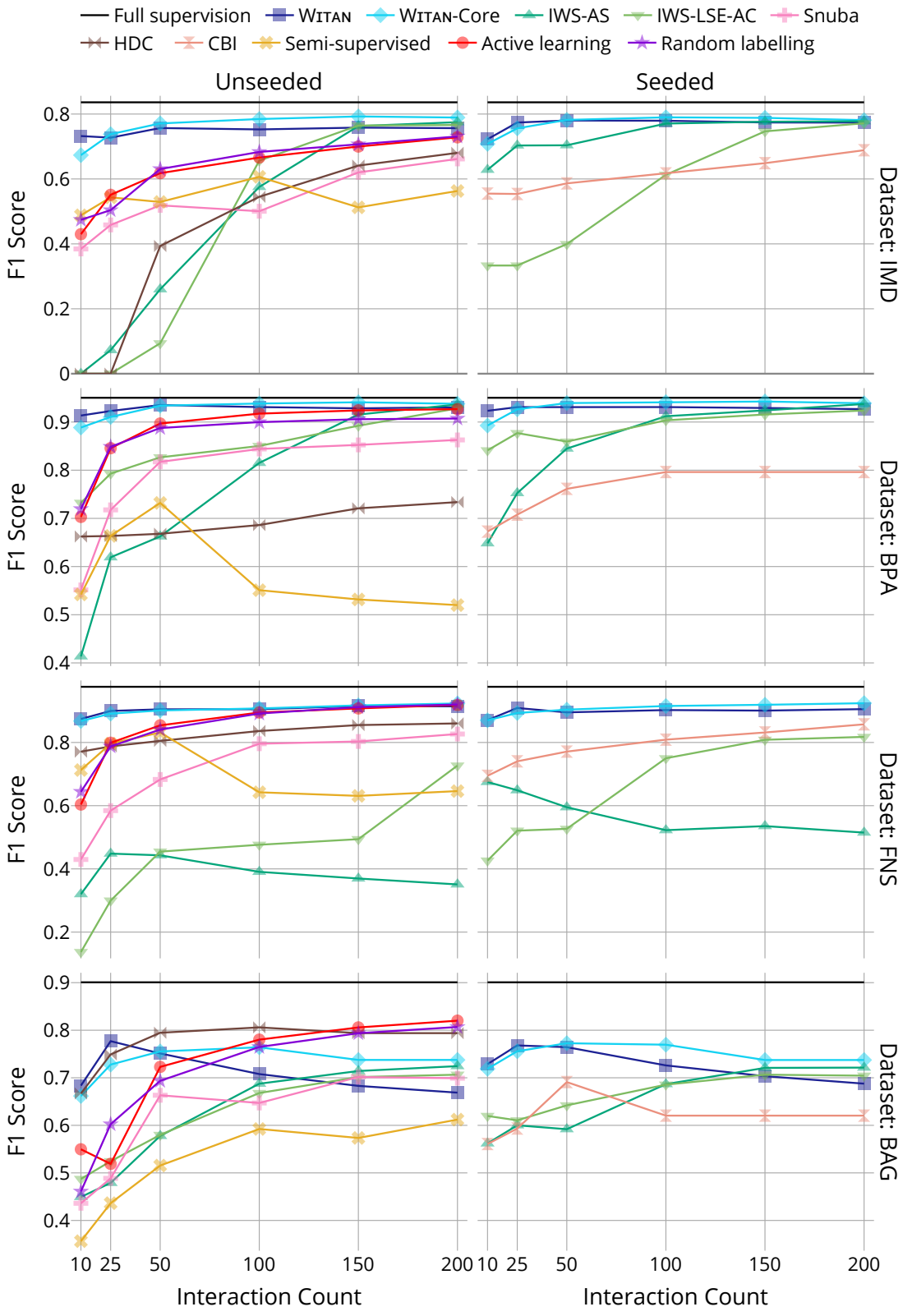


Figure 2.4: Binary classification results.

2.5.3 Runtime Study

Table 2.4 compares the runtimes of labelling methods after 100 interactions, with methods faster and slower than WITAN highlighted in blue and red, respectively. As the computational complexity of WITAN does not depend on the number of classes, runtimes are only presented for binary-class datasets. Without extending the set of candidate LFs with conjunctions and disjunctions, WITAN-Core is slightly faster than WITAN. CBI, Snuba, and semi-supervised learning are also faster in some cases, as their runtimes are not heavily dependent on the number of user interactions. WITAN is generally faster than IWS and active learning, except for the FNS dataset. WITAN is much slower than other methods on FNS due to its large number of features m, as WITAN's runtime scales with m^2. Except for this extreme case, WITAN's maximum average runtime per user interaction is less than 4 seconds (for IMD), making it practical for use in an interactive setting.

2.5.4 Multi-Class Experiments

Fig. 2.5 compares labelling methods on four multi-class classification tasks. All methods are unseeded, except for CBI, which requires seed LFs for two random classes to train its "residual" classifier.

Of the two multi-class extensions of IWS proposed in Section 2.4, IWS-Distinct is superior across all four datasets and performs better with the LSE-AC acquisition strategy in three datasets. In contrast to the binary classification results, WITAN-Core begins to outperform WITAN at lower IC. Furthermore, methods HDC, active learning, semi-supervised learning, and even random labelling are stronger performers, especially at higher IC. These results suggest that the smaller classes of multi-class problems are better captured by lower coverage LFs (e.g. those generated by HDC and WITAN-Core, without disjunctions) or by labelling specific example instances (e.g. active and

Table 2.4: Runtime in seconds for labelling methods. Runtimes are highlighted when slower than WITAN by more than 10 or 60 seconds or faster than WITAN by more than 10 or 60 seconds.

Method	IC	IMD	IMG	BPA	BPT	BJP	BPP	AZN	YLP	PLT	FNS	BDB	BAG	ATW	DMG	SPM
WITAN	100	310.4	173.8	17.0	23.7	23.0	50.1	105.8	98.0	0.2	1576.0	43.8	19.4	0.9	1.3	0.2
WITAN-Core	100	333.2	187.6	13.3	19.2	19.0	26.4	57.3	93.6	0.3	1340.0	21.9	12.7	0.4	0.7	0.2
IWS-AS	100	626.0	559.9	612.7	621.2	617.0	620.8	668.9	632.8	577.6	653.6	641.3	636.2	583.0	618.2	472.5
IWS-LSE-AC	100	655.7	584.3	635.3	640.8	636.1	652.3	690.6	653.0	596.0	673.5	665.6	655.9	602.1	642.5	497.3
Snuba	100	158.7	144.4	48.6	48.6	45.0	46.7	58.6	95.0	14.7	326.2	41.8	31.8	14.9	20.3	3.9
HDC	100	423.4	250.7	11.0	19.1	22.0	19.9	83.8	133.5	0.3	1019.2	34.4	20.6	0.3	0.9	0.2
CBI	100	70.4	55.3	7.1	12.7	15.3	20.7	118.6	34.2	0.7	115.2	26.0	17.9	2.3	2.4	1.4
Semi-supervised	100	30.5	11.2	1.0	4.4	6.4	20.7	550.7	12.9	0.0	33.8	34.3	19.4	0.6	0.1	0.1
Active learning	100	927.8	682.0	203.5	272.4	303.1	355.1	987.5	432.5	40.9	1132.8	459.7	391.6	74.8	102.8	55.8

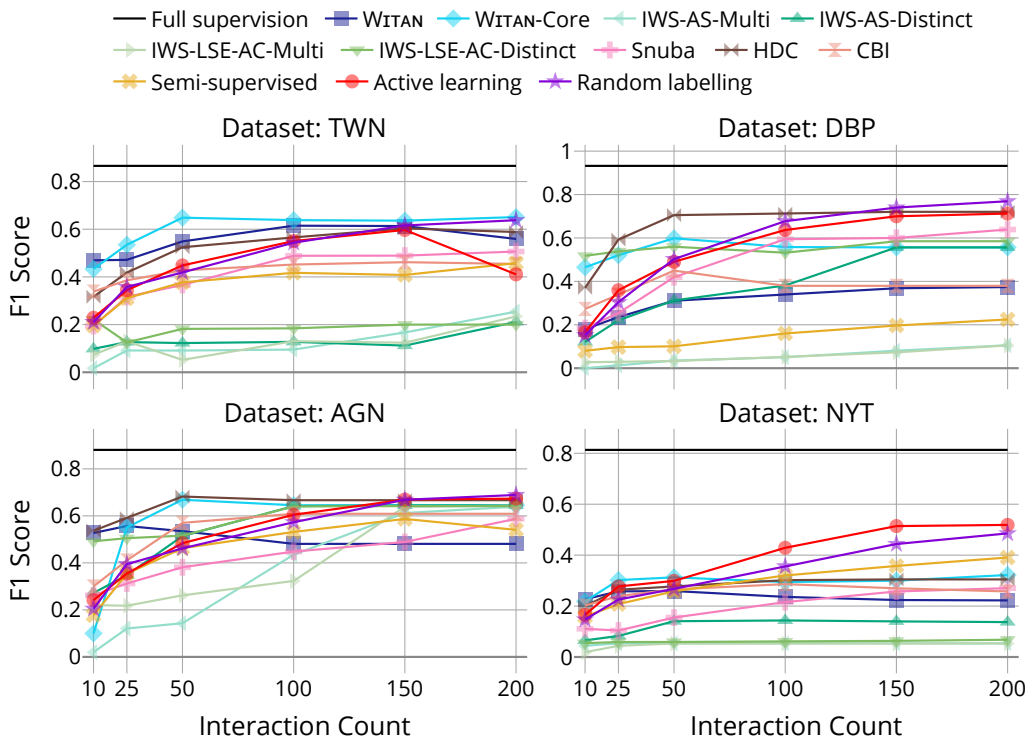


Figure 2.5: Multi-class classification results.

semi-supervised learning and random labelling). An extreme example is the highly-imbalanced NYT dataset, where the smallest class contains only ~ 1% of instances. However, it must be re-emphasised that the instance-labelling methods do not produce interpretable LFs, and HDC LFs are difficult to interpret due to their many features. Overall, WITAN-Core is still generally the best performer up to a moderate interaction count ($IC \leq 100$).

2.5.5 Ablation Study

Table 2.5 presents the results of an ablation study comparing variants of WITAN across binary-class and multi-class datasets, highlighting methods that achieve an F1 score greater or less than WITAN in blue and red, respectively.

These results again highlight that while WITAN outperforms WITAN-Core at low IC for binary classification, WITAN-Core is superior at higher IC and for multi-class

Table 2.5: F1 scores of WITAN variants. Scores highlighted when less than baseline WITAN by more than 0.01 or 0.05, or greater than baseline WITAN by more than 0.01 or 0.05.

Method	IC	IMD	IMG	BPA	BPT	BJP	BPP	AZN	YLP	PLT	FNS	BDB	BAG	ATWDMG	SPMTWN	DBP	AGN	NYT
WITAN	25	0.727	0.710	0.923	0.848	0.892	0.860	0.772	0.737	0.675	0.900	0.979	0.777	0.510	0.723	0.816	0.472	0.235	0.557	0.258
	100	0.753	0.768	0.931	0.817	0.887	0.875	0.779	0.795	0.624	0.905	0.949	0.708	0.550	0.690	0.781	0.615	0.341	0.481	0.236
Core	25	0.738	0.703	0.910	0.836	0.859	0.840	0.760	0.737	0.667	0.891	0.978	0.728	0.632	0.783	0.745	0.535	0.520	0.551	0.302
	100	0.785	0.771	0.938	0.833	0.900	0.848	0.834	0.801	0.719	0.908	0.976	0.764	0.594	0.846	0.843	0.638	0.558	0.645	0.294
No ANDs	25	0.772	0.729	0.936	0.858	0.897	0.877	0.572	0.737	0.675	0.906	0.973	0.775	0.501	0.734	0.816	0.537	0.223	0.507	0.256
	100	0.783	0.770	0.932	0.781	0.893	0.887	0.572	0.808	0.675	0.925	0.973	0.758	0.501	0.732	0.796	0.614	0.223	0.507	0.252
No ORs	25	0.741	0.703	0.910	0.836	0.859	0.830	0.760	0.737	0.667	0.891	0.978	0.728	0.645	0.783	0.745	0.542	0.529	0.551	0.300
	100	0.784	0.767	0.940	0.835	0.900	0.848	0.827	0.781	0.601	0.908	0.977	0.764	0.552	0.820	0.769	0.640	0.562	0.647	0.272
$\gamma = 1$	25	0.727	0.719	0.917	0.842	0.869	0.888	0.705	0.791	0.719	0.891	0.976	0.719	0.661	0.549	0.624	0.512	0.237	0.488	0.250
	100	0.744	0.746	0.928	0.833	0.865	0.901	0.777	0.782	0.684	0.907	0.969	0.722	0.554	0.449	0.624	0.572	0.277	0.477	0.255
Feedback	25	0.770	0.636	0.917	0.820	0.880	0.863	0.744	0.782	0.697	0.837	0.976	0.794	0.502	0.801	0.767	0.533	0.259	0.642	0.253
	100	0.754	0.683	0.922	0.788	0.863	0.882	0.842	0.783	0.657	0.818	0.974	0.753	0.494	0.702	0.738	0.514	0.330	0.633	0.244

datasets. From the *No ANDs* and *No ORs* variants that respectively disable conjunctive and disjunctive LFs, it can be seen that these differences are largely the result of disjunctive LFs. This confirms the earlier observation that WITAN-Core performs better in settings where narrower, more targeted LFs without disjunctions are more appropriate. The generally inferior results with $\gamma = 1$ justify the default setting of $\gamma = 2$ to prefer high information gain over wide coverage for LFs.

Finally, WITAN was tested with interactive user feedback: $f^{\lambda} = \texttt{approved}$ is set for each generated LF λ if it is accurate enough to be assigned a class label by the simulated user. If an LF is not approved, conjunctive child LFs are not added to the candidate set. There is a substantial performance benefit with interactive feedback for some datasets. However, the performance drop for other datasets highlights the importance of exploring LFs that are not necessarily similar to already approved LFs in order to achieve diversity.

2.6 Conclusion

This chapter has addressed RO1 by proposing WITAN, a novel algorithm for the unsupervised generation of labelling functions (LFs) for data programming. The main advantage of WITAN is its support for a variety of user interaction modes, most notably being able to effectively generate LFs without any interactive feedback, labelled instances, seed LFs, or even a predefined set of classes. Experimental results demonstrated that WITAN can be used to train a classifier that is competitive with alternative labelling methods on both binary and multi-class classification tasks, even without initial supervision. This chapter also proposed and evaluated two new extensions to the previously proposed Interactive Weak Supervision (IWS; Boecking et al., 2021) LF generation method that enable its application to multi-class classification tasks.

Chapter 3

Reliable Quantification with Constrained Error Under Unknown Dataset Shift

3.1 Introduction

This chapter is based on the work presented in "Gain-Some-Lose-Some: Reliable Quantification Under General Dataset Shift" (Denham et al., 2021) and "Dynamic Quantification with Constrained Error Under Unknown General Dataset Shift" (Denham et al., 2022a), and addresses RO2:

RO2 Develop a methodology for reliable quantification under unknown dataset shift with constrainable prediction intervals, achieved by the following sub-objectives:

RO2.1 Develop a quantification method that can be safely applied under weaker assumptions of dataset shift than existing quantification methods.

RO2.2 Develop an approach for dynamically selecting an appropriate quantification method for a given target sample in order to exceed the performance of a

single, static quantification method.

RO2.3 Develop a framework for constraining quantification prediction intervals to user-specified limits by reducing uncertainty through requests for ground truth labels from the user for minimal sets of target instances.

Dataset shift remains one of the most significant challenges to deploying supervised machine learning systems (J. Lu et al., 2018; Takahashi & Braga, 2020; Lemberger & Panico, 2020). Applying a model to a population that differs from the one on which it was trained often has an adverse impact on its performance (Rabanser et al., 2019; Takahashi & Braga, 2020; Lemberger & Panico, 2020). Such dataset shifts frequently occur in practice when available training data are a biased sample of the target population, when a model is applied to a new population, or when the target population changes over time (aka *drift* in a stream mining context) (Takahashi & Braga, 2020). *Domain adaptation* methods can be used to adapt models under certain types of shift without additional labelled instances from the target population (Lemberger & Panico, 2020). However, if the expected nature of a shift is not well-understood, then shift-detection methods must be applied to identify if the model should be re-trained with new training instances (Rabanser et al., 2019).

Accounting for dataset shift is particularly important when training *quantification* models. Quantification is a supervised learning task similar to classification, but instead of predicting the class Y of an instance given its features X, the goal is to estimate the proportion of each class within a sample of instances (González et al., 2017). Quantification has a wide range of applications, from comparing sentiment across groups of documents (Keith & O'Connor, 2018; Moreo & Sebastiani, 2022) to monitoring plankton populations across water samples (González et al., 2017, 2019). At first, quantification may appear to simply require applying a classifier and counting predicted

classes (the so-called *classify-and-count* (CC) method), and many quantification methods do rely on an underlying classifier (González et al., 2017). However, as the class distribution is expected to change between training and target samples, shift is inherent to quantification tasks. Specialised quantification methods can exceed CC accuracy by accounting for shift using approaches that crossover with the field of domain adaptation (González et al., 2017). Unfortunately, many quantification methods make strong assumptions about the nature of shift (Saerens, Latinne & Decaestecker, 2002; Forman, 2008; Bella, Ferri, Hernández-Orallo & Ramirez-Quintana, 2010; Keith & O'Connor, 2018; Vaz, Izbicki & Stern, 2019; Alexandari, Kundaje & Shrikumar, 2020; Hassan, Maletzke & Batista, 2020). When those assumptions do not hold, such methods may be less accurate than even CC and produce severely underestimated prediction intervals for estimates.

This chapter presents the *Gain-Some-Lose-Some* (GSLS) model for quantification. GSLS can be safely applied under more general conditions of dataset shift than those assumed by existing quantification methods, including shifts between batches from a data stream. GSLS models shift as the addition or removal, or both addition and removal, of distinct sub-populations with arbitrary class priors and feature distributions. Without any assumptions about a shift's nature, improving a model's accuracy is intractable (Alexandari et al., 2020). Therefore, instead of aiming to improve accuracy, GSLS focuses on providing reliable prediction intervals that accurately reflect potential changes to class priors caused by measured shift. Fig. 3.1 compares GSLS to two prominent quantification methods[1]: probabilistic classify-and-count (PCC; Bella et al., 2010; Keith & O'Connor, 2018) and expectation maximisation (EM; Saerens et al., 2002). PCC is similar to CC in that it assumes no shift, while EM assumes a specific kind of shift known as *prior shift*. All of these methods rely on per-instance probabilistic

[1]Fig. 3.1 presents 80% prediction intervals from experiments for a single random seeding with the ISX dataset, as described in Section 3.6.

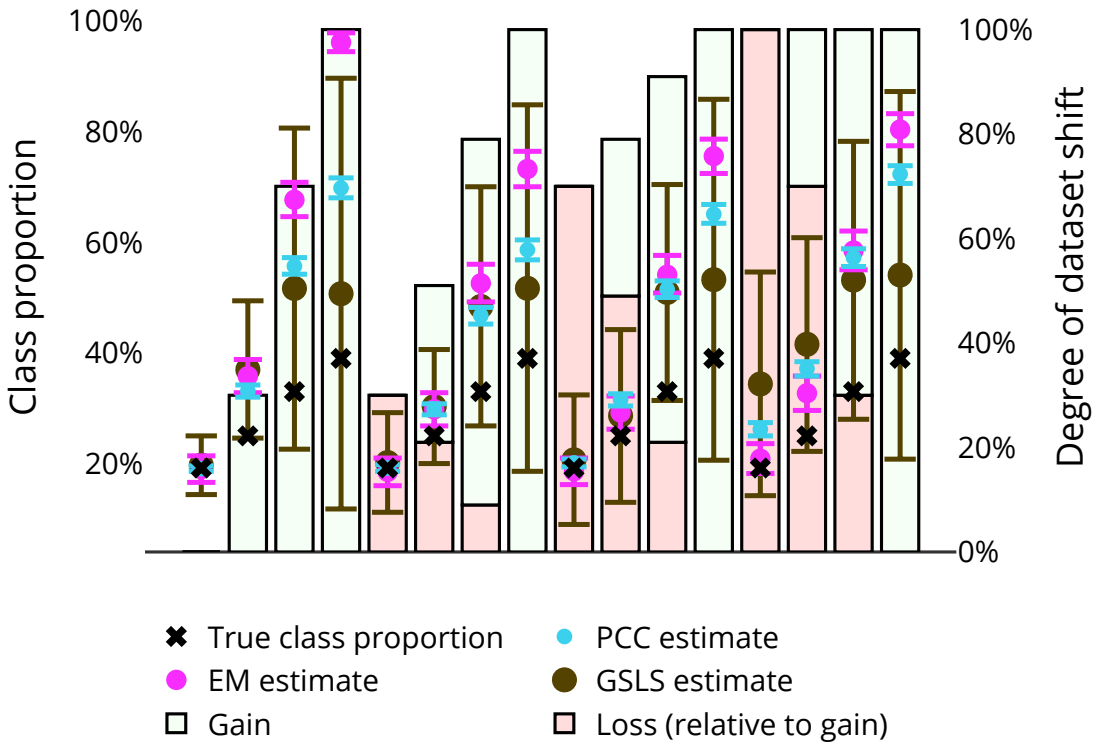


Figure 3.1: Quantification prediction intervals under varied shift.

classifications, which were provided by a logistic regression classifier. As the degree of dataset shift (under the model described in Section 3.6) increases in Fig. 3.1, PCC and EM estimates diverge from the true class proportion while their prediction intervals remain narrow. On the other hand, GSLS intervals widen proportionally with measured shift, providing a more realistic evaluation of the estimate's accuracy.

In addition to presenting the GSLS model, this chapter also incorporates GSLS into a method for dynamically selecting the most appropriate quantification method to account for observed dataset shift when the true shift conditions are not known a priori. Specifically, a statistical testing framework is presented for selecting PCC under no shift, EM under prior shift, and GSLS under any other more general condition of shift.

However, reliable quantification intervals alone may not fully satisfy a user's requirements if they are too wide to provide meaningful insights. While any prediction interval

communicates the degree of potential error better than the binary output of a simple shift detector (Rabanser et al., 2019), neither method instructs the user in what remedial actions are necessary to achieve their desired level of certainty. In order to address this issue, a framework is proposed for constraining quantification interval widths to within user-specified limits. Quantification uncertainty is reduced by requesting class labels from the user for a minimal subset of instances from the target sample.

To summarise the contributions made in this chapter:

1. The *Gain-Some-Lose-Some* (GSLS) model for reliable quantification under general dataset shift (Section 3.3).
2. A decision tree for selecting the most appropriate quantification method based on previously proposed statistical tests for detecting shift and a novel proposal for efficiently detecting non-prior shift (Section 3.4).
3. A framework for constraining quantification intervals to user-specified limits by requesting minimal target instance class labels from the user (Section 3.5).
4. Empirical experiments on benchmark datasets under varied conditions of shift and a real-world dataset previously used to study dataset shift (González et al., 2019) (Section 3.6).

Together, these contributions enable users to apply the most suitable method to achieve reliable quantification with constrained prediction interval widths under unknown conditions of dataset shift.

As this chapter introduces a large amount of notation, the following listing summarises the key terms:

X, Y Random variables for features and classes.

P^S, P^T Source and target population distributions.

P^+, P^-, P^R 'Gain', 'loss', and 'remaining' components of the GSLS model.

w^+, w^- Mixture weights of 'gain' and 'loss'.

$\hat{w}^+$, $\hat{w}^-$ Estimates of mixture weights.

$\{X^D, Y^D\}^{n^D}$ A sample of n^D instance features and classes from distribution D.

h^D Histogram of distribution D. A histogram's bin count is typically referred to as b.

q_c^D Proportion of instances in a sample from distribution D belonging to class c.

$\hat{q}_c^D$ Quantification estimate of q_c^D.

$f(x)$ Classifier that predicts a class value for instance features x.

$g_c(x)$ Probabilistic classifier that predicts $P(Y = c|x)$ for instance features x.

$V(X)$ Variance of some random variable X.

$I(a)$ Indicator: $I(a) = 1$ if a is true, else 0.

3.2 Literature Review

This section reviews the different types of dataset shift and approaches to handling them, as well as previously proposed quantification methods.

3.2.1 Types of Dataset Shift

Dataset shift is typically defined as any difference in the joint distribution of the features X and class Y between the source population S that a model is trained on and the target population T it is applied to: $P^S(X, Y) \neq P^T(X, Y)$ (Takahashi & Braga, 2020). This definition will be referred to as *general shift* to distinguish it from the key sub-types of dataset shift discussed below.

Domain adaptation methods that do not require labelled instances from the target population primarily target two types of shift: *covariate shift* and *prior shift*. Covariate shift occurs when the feature distribution changes ($P^S(X) \neq P^T(X)$) but posterior class probabilities are invariant ($P^S(Y|X) = P^T(Y|X)$) (Takahashi & Braga, 2020). A model can account for covariate shift by re-weighting training instances according to the ratio of feature probabilities: $P^T(X)/P^S(X)$ (Lemberger & Panico, 2020).

Prior shift occurs when class priors change ($P^S(Y) \neq P^T(Y)$) but class-conditional feature distributions are invariant ($P^S(X|Y) = P^T(X|Y)$) (Vaz et al., 2019). If target class priors $P^T(Y)$ are known, the predictions of a probabilistic classifier can be directly adjusted to prior shift (Saerens et al., 2002). Because $P^T(Y)$ is almost always unknown in practice, the problem of adaptation under prior shift can be addressed by the EM quantification method discussed in the next subsection. Quantification methods commonly assume prior shift, but there are many applications where the prior shift assumption will not hold. Consider quantifying classes of failure in a stream of product warranty claims. If an underlying cause of failure is resolved, unknown sub-populations of related failure classes may disappear from the target population. Similarly, new faults may introduce previously unseen sub-populations across multiple failure classes. Either situation may alter both $P(Y)$ and $P(X|Y)$.

Some domain adaptation methods rely on relaxations of the prior and covariate shift assumptions, such as assuming the invariance of feature distributions for inferred subclasses of target classes (Alaíz-Rodríguez, Guerrero-Curieses & Cid-Sueiro, 2011), or by seeking a feature transformation ϕ that preserves invariance of posterior class probabilities: $P^S(Y|\phi(X)) = P^T(Y|\phi(X))$ (Bouvier, Very, Hudelot & Chastagnol, 2019). These assumptions still exclude important types of shift, such as the introduction of an entirely new sub-population with distribution $P(X, Y)$ unrelated to $P^S(X, Y)$. One-class quantification methods (D. dos Reis, Maletzke, Cherman & Batista, 2018) can account for the addition of an entirely new distribution in the target population but assume that it will not contain instances belonging to the one target class. Some domain adaptation methods designed for stream mining, including some quantification methods (Hofer, 2015; Moreira dos Reis, Maletzke, Silva & Batista, 2018; Das, Lade & Srinivasan, 2020), assume gradual drift or recurring concepts over time. However, these methods are not designed to handle abrupt shifts to unknown concepts.

Without any assumptions on the nature of a shift, domain adaptation to improve a model's accuracy is intractable (Alexandari et al., 2020). In these challenging cases, unsupervised shift detection can still identify potentially detrimental shifts that necessitate model re-training on new labelled instances from the target population. Such methods have been developed to detect changes in the distributions of features and probabilistic classifications (Cieslak & Chawla, 2009; Rabanser et al., 2019), and some test whether certain types of shift (e.g. prior shift) are likely to be present (A. Maletzke, Reis, Cherman & Batista, 2018; Vaz et al., 2019). However, unsupervised shift detection cannot detect shift when posterior class probabilities change ($P^S(Y|X) \neq P^T(Y|X)$) while observed feature distributions remain invariant ($P^S(X) = P^T(X)$). This specific type of shift, known as *concept shift*, can only be detected by collecting labelled instances from the target population or under assumptions of gradual drift in a stream (Žliobaite, 2010; D. M. dos Reis, 2020; Lemberger & Panico, 2020).

As the proposed GSLS model is designed for cases of general shift without additional assumptions, its aim is similar to that of unsupervised shift detection: identifying potentially detrimental shifts even when model improvement is intractable. However, GSLS also measures the potential impact of identified shift on quantification, expressed as a prediction interval. Only one other known work models the impact of an unknown general shift: a bound for classification error based on the divergence between source and target feature distributions (measured by a classifier trained to separate the distributions) (Ben-David et al., 2010).

3.2.2 Quantification Methods

To formally define the problem setting for quantification, let $P^S(X, Y)$ and $P^T(X, Y)$ represent distributions of the source S and target T populations over an m-dimensional feature space $X \in \mathbb{R}^m$ and k classes $Y \in \{c_1, c_2, \ldots, c_k\}$. The goal of quantification is to

use a complete source sample $\{X^S, Y^S\}^{n^S} \sim P^S(X,Y)$ and only the features X^T of a target sample $\{X^T, Y^T\}^{n^T} \sim P^T(X,Y)$ to estimate the proportions of target instances having each class:

$$q_c^T = \frac{1}{n^T} \sum_{i=1}^{n^T} I\left(y_i^T = c\right) \quad \forall \quad c \in Y \tag{3.1}$$

where I is the indicator function. González et al. (2017) thoroughly review quantification methods for estimating q_c^T, but this research focuses on methods that meet two criteria. Firstly, only methods that are based on the outputs of a classifier are reviewed, as they have been found to be of more practical value than special-purpose quantification learners (Tasche, 2016). Secondly, given this research's goal of providing reliable prediction intervals around quantification estimates, only methods that can provide such intervals are reviewed.

The first quantification method of interest is *probabilistic classify-and-count* (PCC) (Bella et al., 2010; Keith & O'Connor, 2018). PCC is a variation of the standard classify-and-count (CC) method that counts the classes assigned to the target sample by a classifier $f(X) \rightarrow Y$:

$$\hat{q}_c^{T\{CC\}} = \frac{1}{n^T} \sum_{i=1}^{n^T} I\left(f\left(x_i^T\right) = c\right) \tag{3.2}$$

Instead of using "hard" classifier outputs, PCC sums the "soft" outputs of a probabilistic classifier g, which approximate posterior class probabilities ($g_c(x) \approx P^S(y = c|x)$):

$$\hat{q}_c^{T\{PCC\}} = \frac{1}{n^T} \sum_{i=1}^{n^T} g_c\left(x_i^T\right) \tag{3.3}$$

As the approximate posterior probabilities are trained on the source sample, PCC assumes there is no shift between the source and target distributions ($P^S(X,Y) = P^T(X,Y)$) (Forman, 2008). Despite this extreme assumption, PCC is still relevant here because it was recently noted that $\hat{q}_c^{T\{PCC\}}$ represents the mean of a Poisson binomial

(PB) distribution of the posterior probabilities scaled by $1/n^T$ (Keith & O'Connor, 2018). Therefore, a credible interval of this distribution is a prediction interval[2] for $\hat{q}_c^{T\{PCC\}}$.

Expectation maximisation (EM) quantification is the second quantification method of interest (Saerens et al., 2002). EM assumes prior shift, relying on $P^S(X|Y) = P^T(X|Y)$ to iteratively update class prior and posterior probabilities (from a probabilistic classifier) to maximise the likelihood of target features X^T:

$$\begin{aligned}
\hat{P}^{T(0)}(c) &= \hat{P}^S(c) \\
\hat{P}^{T(s)}\left(c|x_i^T\right) &= \frac{\frac{\hat{P}^{T(s)}(c)}{\hat{P}^S(c)} g_c(x_i^T)}{\sum_{d \epsilon Y} \frac{\hat{P}^{T(s)}(d)}{\hat{P}^S(d)} g_d(x_i^T)} \\
\hat{P}^{T(s+1)}(c) &= \frac{1}{n^T} \sum_{i=1}^{n^T} \hat{P}^{T(s)}\left(c|x_i^T\right) \\
\hat{q}_c^{T\{EM\}} &= \hat{P}^{T(t)}(c)
\end{aligned} \tag{3.4}$$

where t represents the iterations required for convergence, and $\hat{P}^S(c)$ is an estimate of the source class prior given by the mean of $g_c(x)$ over a held-out portion of the source sample (in order to avoid the effects of overfitting; Alexandari et al., 2020). This method has been widely used in the quantification literature, referred to as EM (Lemberger & Panico, 2020; Vaz et al., 2019), maximum-likelihood (ML; Tasche, 2019), and SLD (Esuli, Molinari & Sebastiani, 2020). Because EM produces maximum-likelihood estimates of class priors $P^T(Y)$, the Fisher information can be used to produce a confidence interval (Saerens et al., 2002; Keith & O'Connor, 2018; Tasche, 2019)[3]. Tasche (2019) noted that a confidence interval for $P^T(Y)$ is distinct from a prediction

[2] Keith and O'Connor (2018) described these as confidence intervals, but they were later noted to be prediction intervals by Tasche (2019).

[3] Saerens et al. (2002) state that "standard errors on the estimated a priori probabilities can be obtained through the computation of the observed information matrix", Tasche (2019) notes that "an asymptotically most efficient normal approximation with variance expressed in terms of the Fisher information can be used", and Keith and O'Connor (2018) use the negative curvature of the marginal log-likelihood, which is the same as the Fisher information.

interval for a sample class proportion q_c^T but found that confidence intervals can still provide adequate coverage of realised class proportions. Tasche (2017) has also shown that EM is a Fisher consistent estimator, even under a minor relaxation of the prior shift assumption (referred to as the "invariant density ratio" assumption) that allows for certain kinds of covariate shift. EM is a popular and effective method (Esuli et al., 2020) and is the only member of the competitive family of distribution-matching quantification methods (A. Maletzke, Hassan, Reis & Batista, 2020; Schumacher, Strohmaier & Lemmerich, 2021) that can natively provide a confidence or prediction interval.

Approaches to producing confidence intervals for other quantification methods have also been proposed. Vaz et al. (2019) prove a central limit theorem for the "ratio estimator" family of methods, but they only demonstrate this for binary-class problems. Daughton and Paul (2019) use bootstrapping to produce confidence intervals for quantification estimates, but these were later shown to fail under prior shift (Tasche, 2019). Tasche (2019) uses bootstrapping to produce confidence and prediction intervals for any quantification method and finds these intervals to be more reliable than native EM intervals on small target samples. However, the proposed method requires re-applying the quantification method (and potentially re-training the underlying classifier) for each bootstrapped sample, which will be prohibitively expensive when quantifying many target samples.

As PCC and EM are the only identified quantification methods that natively produce a prediction or confidence interval in a multi-class setting, they are the state-of-the-art methods for producing quantification intervals to be compared with the GSLS model. Note that each of these methods is designed to be the best performer under a particular assumption of shift: PCC assumes no shift, EM assumes prior shift, and GSLS only assumes general shift. Therefore, Section 3.4 considers how to detect each shift condition in order to select the most appropriate of the three methods, and

Section 3.5 presents methods for constraining the quantification intervals of all three methods. Note that PCC and EM, as well as the GSLS model, assume classification outputs are produced by a probabilistic classifier and therefore approximate posterior probabilities $P^S(Y|X)$.

3.3 Proposed GSLS Model of General Shift

Unlike existing quantification methods that place strong assumptions on the nature of dataset shift, the proposed *Gain-Some-Lose-Some* (GSLS) model produces quantification estimates with reliable prediction intervals under general dataset shift. Fig. 3.2 demonstrates how GSLS expresses any general shift between the source and target distributions P^S and P^T. GSLS is named after its two constituent transformation steps:

1. A subset of the source population with arbitrary distribution P^- may be removed in the target population. P^S is modelled as a mixture of this 'loss' distribution P^- and the complement 'remaining' distribution P^R (which *remains* in the target distribution) weighted by 'loss weight' $w^- \in [0, 1]$: $P^S = w^- P^- + (1 - w^-)P^R$. This can be viewed as the 'loss' of a concept from P^S, e.g. resolving a fault that was causing an unknown distribution of failure classes in warranty claims.
2. A new population with arbitrary distribution P^+ that was not present in the source population may be introduced in the target population. P^T is modelled as a mixture of this 'gain' distribution P^+ and P^R weighted by 'gain weight' $w^+ \in [0, 1]$: $P^T = w^+ P^+ + (1 - w^+)P^R$. This can be viewed as 'gaining' a new concept in P^T, e.g. a new fault that causes an unknown distribution of failure classes in warranty claims.

As P^-, P^+, and P^R are all independent distributions, each transformation can represent any general dataset shift $P^S \neq P^T$. For example, when $w^- = 1$ or $w^+ = 1$,

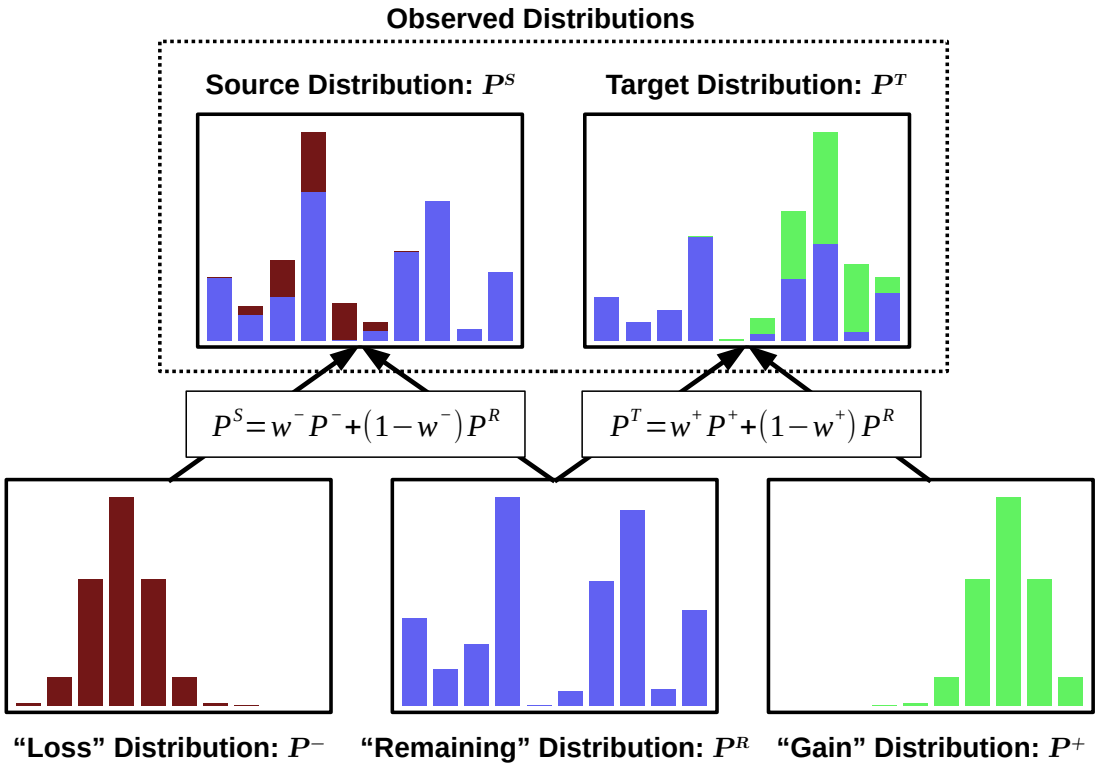


Figure 3.2: Gain-Some-Lose-Some (GSLS) Model.

P^S and P^T will be completely independent. Non-exhaustive examples of specific shift types are also seen in special cases of GSLS:

- The prior shift assumption of $P^S(X|Y) = P^T(X|Y)$ will hold when the class-conditional distributions are invariant across all of the components: $P^+(X|Y) = P^-(X|Y) = P^R(X|Y)$.
- The covariate shift assumption of $P^S(Y|X) = P^T(Y|X)$ will hold when posterior class probabilities are invariant across all of the components: $P^+(Y|X) = P^-(Y|X) = P^R(Y|X)$.
- Concept shift occurs when the components have identical distributions of features ($P^+(X) = P^-(X) = P^R(X)$) but have different posterior class probabilities: $P^+(Y|X) \neq P^-(Y|X) \neq P^R(Y|X)$.

The following subsections discuss the two steps required to apply the GSLS model in the context of quantification:

1. Fitting GSLS to source and target sample distributions. Distribution representations must be based on features X, as classes Y are not observed for the target sample (Section 3.3.1).
2. Producing a reliable quantification prediction interval that accounts for the potential impact of the unknown 'gain' and 'loss' class distributions $P^{-}(Y)$ and $P^{+}(Y)$ (Section 3.3.3).

Conditions under which the proposed fitting approach may underestimate shift are also discussed in Section 3.3.2.

3.3.1 Fitting the GSLS Model

The following section presents a concrete method for fitting the GSLS model to given source X^S and target X^T feature samples.

Representation of Distributions

In order to fit GSLS, an appropriate representation for distributions of features X must be selected. As X can be high-dimensional in many application domains, directly comparing distributions of X will suffer from the 'curse of dimensionality' (Jain, White, Trosset & Radivojac, 2016). Therefore, GSLS is fit using the common approach (Cieslak & Chawla, 2009; Jain et al., 2016; Rabanser et al., 2019; Alexandari et al., 2020; Vaz et al., 2019) of working with distributions of outputs from a classifier trained on the source sample: $g_c(X)$. In order to avoid the effects of overfitting, the source distribution is produced from a held-out 'calibration' portion of the source sample that was not used for classifier training.

Making no assumptions about the shape of $g_c(X)$ distributions, they are represented with histograms over the range of possible values $[0, 1]$. The number of bins b is an important hyperparameter: If $b = 1$, all distributions will appear identical, but extremely high values of b will make even similar distributions appear very different. Fortunately, sensitivity to b can be minimised by using equiprobable bins (Prins, McCormack, Michelson & Horrell, 2002) such that each bin has an equal number of source sample instances, making differences in the target distribution clearly visible. Setting $b = 2\min(n^S, n^T)^{2/5}$ is recommended for Chi-square goodness-of-fit tests with equiprobable bins (Prins et al., 2002), and this rule-of-thumb is found to be a reasonable setting in the experiments presented in Section 3.6.

To summarise, each distribution D is represented with a histogram ($h^D \in [0, 1]^b$, such that $\sum_{i=1}^{b} h_i^D = 1$) of $g_c(X)$ values with b bins that are equiprobable for source distribution S.

Optimisation Criterion for Fitting GSLS

When fitting the GSLS model for any given source and target distributions, there are infinitely many possible shapes and weights for P^+, P^-, and P^R. Therefore, a criterion is required to select best-fit parameter values. In order to avoid assumptions of distribution shapes, the chosen criterion is to explain the observed P^S and P^T **with the minimal degree of gain and loss shift**. Formally, the sum of estimated gain and loss weights is minimised: $\hat{w}^+ + \hat{w}^-$. This criterion reflects real-world applications, as there is little merit in applying a model of P^S to a sample of P^T if their differences outweigh their similarities.

An alternative criterion was also considered: maximising the weight of P^R shared by P^S and P^T: $(1-\hat{w}^+)(1-\hat{w}^-)$. The key difference is that this criterion is biased towards balanced, moderate loss and gain (e.g. $\hat{w}^+ = 0.5, \hat{w}^- = 0.5$) over unbalanced loss and gain (e.g. $\hat{w}^+ = 0.1, \hat{w}^- = 0.9$). Initial experimentation found that this criterion resulted

in more underestimation of P^R's weight under certain shift conditions. Conditions under which the minimal shift criterion may underestimate shift are discussed in Section 3.3.2.

Fitting GSLS for Minimal Shift

To fit GSLS for minimal $\hat{w}^+ + \hat{w}^-$ given source and target histograms h^S and h^T, first consider starting with a fixed histogram for the remaining distribution h^R. As all possible shapes for h^+ and h^- are assumed to be equally likely, shapes are selected to minimise their weights in the mixtures for h^S and h^T respectively. This can be achieved by minimising $\hat{w}^+$ and $\hat{w}^-$ while ensuring that the scaled h^R still "fits inside" histograms h^S and h^T:

$$\begin{aligned} \left((1-\hat{w}^+)h_i^R \le h_i^T\right) \;\forall\; i \in 0,\ldots,b \\ \left(\hat{w}^+ \ge 1 - h_i^T/h_i^R\right) \;\forall\; i \in 0,\ldots,b \\ \min(\hat{w}^+) = 1 - \left(\min_{i,\, h_i^R>0} \frac{h_i^T}{h_i^R}\right) \\ \text{Similarly: } \min(\hat{w}^-) = 1 - \left(\min_{i,\, h_i^R>0} \frac{h_i^S}{h_i^R}\right) \end{aligned} \tag{3.5}$$

This approach bears similarity to the "One Distribution Inside" method for one-class quantification (D. dos Reis et al., 2018). To avoid division by zero, bins where $h_i^R = 0$ are ignored, as they will fit inside the mixture histograms for any weight. With the minimum values of $\hat{w}^+$ and $\hat{w}^-$, the shapes of h^+ and h^- can be found (for any non-zero weights, otherwise the distribution is non-existent):

$$\begin{aligned} P^T &= w^+P^+ + (1-w^+)P^R \\ \therefore h_i^T &= \hat{w}^+h_i^+ + (1-\hat{w}^+)h_i^R \\ h_i^+ &= \frac{h_i^T - (1-\hat{w}^+)h_i^R}{\hat{w}^+} \\ \text{Similarly: } h_i^- &= \frac{h_i^S - (1-\hat{w}^-)h_i^R}{\hat{w}^-} \end{aligned} \tag{3.6}$$

Given that the minimum weights can be found for a given h^R, the problem of fitting GSLS for minimal shift is simply to select histogram h^R that minimises $\hat{w}^+ + \hat{w}^-$. This can be formulated as the following constrained optimisation problem:

$$\begin{aligned}
&\arg\min_{h^R} \left[\hat{w}^+ + \hat{w}^-\right] \\
&= \arg\min_{h^R} \left[1 - \left(\min_{i,\, h_i^R>0} \frac{h_i^T}{h_i^R}\right) + 1 - \left(\min_{j,\, h_j^R>0} \frac{h_j^S}{h_j^R}\right)\right] \\
&= \arg\min_{h^R} \left[-\left(\min_{i,\, h_i^R>0} \frac{h_i^T}{h_i^R} + \min_{j,\, h_j^R>0} \frac{h_j^S}{h_j^R}\right)\right] \\
&= \arg\max_{h^R} \left[\min_{i,\, h_i^R>0} \frac{h_i^T}{h_i^R} + \min_{j,\, h_j^R>0} \frac{h_j^S}{h_j^R}\right] \\
&\text{Constrained by: } h_i^R \geq 0 \;\forall\; i \in 1, \ldots, b;\; \textstyle\sum_{i=1}^{b} h_i^R = 1
\end{aligned} \tag{3.7}$$

This constrained optimisation problem is solved by applying Sequential Least Squares Programming (SLSQP) as implemented in SciPy (Virtanen et al., 2020). As the target is not fully differentiable, undefined gradients are avoided by setting $\frac{\partial f}{\partial h_i^R} = 0$ when $h_i^S = 0$ or $h_i^T = 0$ and assigning a gradient to all minima when multiple bins share the same minimum value for $\frac{h_i^T}{h_i^R}$ or $\frac{h_j^S}{h_j^R}$. Because the target is not convex, the optimal result achieved over random initialisations of h^R is selected as the final result. A small number of initialisations (10) achieved acceptable results in experiments (Section 3.6).

The optimisation problem can be simplified when $h_i^S = 0$ or $h_i^T = 0$ for any bins i. Note that for any h^S and h^T, setting $h^R = h^T$ gives $\hat{w}^+ = 0$. Since $\hat{w}^- \leq 1$, it can be asserted that $\min_{h^R} \left[\hat{w}^+ + \hat{w}^-\right] \leq 1$. If $h_j^S > 0 \;\forall\; j$ s.t. $h_j^T > 0$, then setting $h^R = h^T$ still allows h^R to 'fit inside' h^S for $\hat{w}^- < 1$, giving $\min_{h^R} \left[\hat{w}^+ + \hat{w}^-\right] < 1$ in this case. Conversely, if $\exists\; i$ s.t. $h_i^S = 0, h_i^R > 0$, then $\min(\hat{w}^-) = 1$ and consequently $\min_{h^R,\, h_i^R>0} \left[\hat{w}^+ + \hat{w}^-\right] = 1$. These properties also hold if the roles of h^S and h^T are reversed, so a global optimum of the target function must exist when $h_i^R = 0 \;\forall\; i$ s.t. $\min(h_i^S, h_i^T) = 0$. In these situations, the large space of gradients towards local optima where $h_i^R > 0$ can be avoided by fixing

$h_i^R = 0$ and removing bins i from the optimisation problem.

While solving the optimisation problem in a multi-class context is theoretically possible by using multi-dimensional histograms $h \in [0, 1]^{b,k}$ over vectors of classifier scores $g(X) = \{g_c(X) \ \forall \ c \in Y\}$, the difficulty and computational cost will increase greatly with k. Therefore, a separate GSLS model is fit and used to quantify each class.

As the cost of fitting GSLS is dominated by optimising the target function (which scales linearly in the number of bins), it has complexity $O(\eta b)$, where η is the number of target function calls in all iterations over all initialisations of the SLSQP solver. The number of function calls can be reduced by directly defining the Jacobians of the target function and equality constraint, which saves the solver from estimating gradients from finite differences. Note that classifier re-training is not needed to fit GSLS to new target samples, unlike the method for bounding classification error under shift proposed by Ben-David et al. (2010) that must train a population-discriminating classifier for each pair of source and target samples.

3.3.2 Limitations of the Minimal Shift Criterion

The following section considers conditions where fitting GSLS to minimise the degree of shift may result in shift underestimation.

The minimal shift criterion will underestimate shift when changes in the true classification rule $P(Y|X)$ are not reflected in corresponding changes to the observed feature distribution $P(X)$. This describes situations where either $P^+(X)$ or $P^-(X)$ is similar to $P^S(X)$. Formally, P^+ or P^- could be considered a mixture distribution with component P^* such that $P^*(X) = P^S(X)$ and $P^*(Y|X) \neq P^S(Y|X)$. P^* represents a case of concept shift, which is completely undetectable without labelled instances from the target population, as previously discussed (Žliobaite, 2010; D. M. dos Reis, 2020; Lemberger & Panico, 2020). For many real-world datasets, it is reasonable to

assume that $P(Y|X)$ changes will be reflected in $P(X)$. For example, it is unlikely that two populations of documents with similar text feature distributions will have different distributions of class labels.

Overlap between the true gain $P^+(X)$ and loss $P^-(X)$ feature distributions will also result in less observable shift between $P^S(X)$ and $P^T(X)$. Formally, the absolute shift between source and target feature distributions $|P^T(X) - P^S(X)| = |(P^T(X) - P^R(X)) - (P^S(X) - P^R(X))|$ is less than the sum of absolute shifts in the gain and loss steps $|P^T(X) - P^R(X)| + |P^S(X) - P^R(X)|$ when the density at a point a increases in one step and decreases in the other: when $P^R(X = a) < P^S(X = a)$ and $P^T(X = a) > P^R(X = a)$, or when $P^R(X = a) > P^S(X = a)$ and $P^T(X = a) < P^R(X = a)$. The result is shift that is undetectable from feature distributions, though future work could estimate the expected overlap between $P^+(X)$ and $P^-(X)$ based on the observed shift and update $\hat{w}^+$ and $\hat{w}^-$ accordingly.

By using a classifier output distribution $P(g_c(x))$ as a one-dimensional projection of $P(X)$, certain changes to $P(X)$ may be hidden in $P(g_c(X))$, resulting in undetectable shift. This is analogous to the observation of Vaz et al. (2019) that prior shift adaptation based on a classifier actually relies on a "weak prior shift assumption": $P^S(g_c(X)|Y) = P^T(g_c(X)|Y)$.

Even when shift is observable from changes in $P(X)$, simultaneous modelling of gain and loss may result in misinterpretation of loss as gain and vice versa. If the overall degree of shift estimated is accurate, the impact on prediction intervals should be marginal. However, the minimal shift criterion may "explain" some observed shifts as a combination of gain and loss with an underestimated degree of shift. If only gain or loss shift is expected, this issue can be avoided by fixing $h^R = h^S$ or $h^R = h^T$ respectively. However, accounting for only gain or loss when both are present may result in shift overestimation. For example, a gain-only model will interpret a small loss in one histogram bin as a large gain across all other bins.

3.3.3 Prediction Intervals for GSLS

In this section, GSLS is applied to produce quantification estimates with reliable prediction intervals. The distribution of quantification target q_c^T (the proportion of target sample instances with class c) can be modelled with a binomial distribution: $q_c^T \sim \frac{1}{n^T} B\left(n^T, P^T(Y = c)\right)$. However, as $P^T(Y)$ is unknown under general shift, q_c^T is instead modelled using knowledge of P^S and GSLS components P^R, P^+, and P^-.

Having modelled P^T as a mixture of P^R and P^+, the probability of q_c^T can be expressed as the probability of a linear combination of q_c^R and q_c^+ (the class proportions in hypothetical samples from those component distributions). Because mixture weight w^+ is directly estimated from source and target samples, it is treated as a fixed value and not a random variable:

$$\begin{aligned} P(q_c^T = t) &= P(w^+ q_c^+ + (1 - w^+) q_c^R = t) \\ &= P\left(q_c^+ = \frac{t - (1 - w^+) q_c^R}{w^+}\right) \\ &= \int_0^1 P\left(q_c^+ = r + \frac{t - r}{w^+} \middle| q_c^R = r\right) P\left(q_c^R = r\right) dr \\ &= \int_0^1 P\left(q_c^+ = r + \frac{t - r}{w^+}\right) P\left(q_c^R = r\right) dr \end{aligned} \tag{3.8}$$

Recall from the definition of the GSLS model that gain distribution P^+ represents the introduction of an unknown sub-population in the target that may be completely unrelated to P^R. This allows $P(q_c^+)$ to be considered independent of $P(q_c^R)$, such that $P(q_c^+ | q_c^R) = P(q_c^+)$ in (3.8). As the distribution of $P(q_c^+)$ is unknown, it is modelled as a uniform distribution[4]: $q_c^+ \sim \mathrm{U}(0, 1)$.

[4] An earlier formulation (Denham et al., 2021) assumed all possible priors for the k classes to be equally likely by modelling the set of class priors with a flat Dirichlet distribution, giving $q_c^+ \sim \mathrm{Beta}(1, k - 1)$. q_c^+ is now modelled with a uniform distribution, giving identical results when $k = 2$, but having less bias towards a balanced class distribution when $k > 2$. Also, the distribution of q_c^- is now similarly based on a uniform distribution.

As P^R is a component in the mixture of known distribution P^S, q_c^R can also be expressed in terms of q_c^S, q_c^-, and w^-:

$$\begin{aligned} P(q_c^R = r) &= P\left(q_c^S = w^- q_c^- + (1 - w^-)r\right) \\ &= P\left(q_c^- = \frac{q_c^S - (1 - w^-)r}{w^-}\right) \\ &= \int_0^1 P\left(q_c^- = r + \frac{s - r}{w^-} \middle| q_c^S = s\right) P\left(q_c^S = s\right) ds \end{aligned} \tag{3.9}$$

The distribution of source sample class proportion q_c^S can be modelled using probabilistic classifications for the target sample: $g_c(X^T)$, which are generated under the assumption that instances are drawn from the source distribution. Therefore, q_c^S can be modelled with the same Poisson binomial (PB) distribution that PCC uses under the assumption of no shift:

$$q_c^S \sim \frac{1}{n^T} \text{PB}\left(g_c(X^T)\right) \tag{3.10}$$

As this distribution is discrete for the n^T instances in sample X^T, it is approximated by a Gaussian distribution with the same mean and variance truncated to probability range $[0, 1]$ to provide a probability for any continuous q_c^S.

Recall that P^- represents an arbitrary sub-population with an unknown class distribution that is lost from P^S to produce P^R. Similar to q_c^+, there is little knowledge on which to model q_c^-. However, by re-arranging the terms of the q_c^S combination to give $q_c^R = \frac{q_c^S - w^- q_c^-}{1 - w^-}$, it can be seen that there is a limited range of q_c^- values that ensure $0 \leq q_c^R \leq 1$ for a given q_c^S and w^-:

$$\begin{aligned} q_c^R &\leq 1 \\ \frac{q_c^S - w^- q_c^-}{1 - w^-} &\leq 1 \\ q_c^- &\geq \frac{q_c^S - 1}{w^-} + 1 \end{aligned} \quad \middle| \quad \begin{aligned} q_c^R &\geq 0 \\ \frac{q_c^S - w^- q_c^-}{1 - w^-} &\geq 0 \\ q_c^- &\leq \frac{q_c^S}{w^-} \end{aligned} \tag{3.11}$$

Therefore, q_c^- is dependent on q_c^S, and q_c^- can be modelled with a uniform distribution truncated to the range of possible values that ensure $0 \leq q_c^R \leq 1$ and $0 \leq q_c^- \leq 1$ for a given q_c^S:

$$q_c^- \sim \mathrm{U}\left(\max\left(0, \frac{q_c^S - 1}{w^-} + 1\right), \min\left(1, \frac{q_c^S}{w^-}\right)\right) \tag{3.12}$$

Taken together, the above formulations can express $P(q_c^T)$ in terms of Poisson binomial distribution $P(q_c^S)$ and uniform distributions $P(q_c^+)$ and $P(q_c^-|q_c^S)$. As q_c^T represents the proportion of class c in the n^T target instances, a discrete distribution can be computed for points $q_c^T = \frac{\tilde{t}}{n^T}\ \forall\ \tilde{t} \in \{0, \ldots, n^T\}$. Replacing integrations with sums and plugging in estimated $\hat{w}^+$ and $\hat{w}^-$ gives the final formulation for computing $P(q_c^T)$:

$$\begin{aligned} P\left(q_c^T = \frac{\tilde{t}}{n^T}\right) &= \sum_{\tilde{r}=0}^{n^{(R)}} \left[\begin{array}{l} P\left(q_c^+ = \frac{\tilde{r}}{n^{(R)}} + \frac{\frac{\tilde{t}}{n^T} - \frac{\tilde{r}}{n^{(R)}}}{\hat{w}^+}\right) \\ \times P\left(q_c^R = \frac{\tilde{r}}{n^{(R)}}\right) \end{array} \right] \\ P\left(q_c^R = \frac{\tilde{r}}{n^{(R)}}\right) &= \sum_{\tilde{s}=0}^{n^{(S)}} \left[\begin{array}{l} P\left(q_c^- = \frac{\tilde{r}}{n^{(R)}} + \frac{\frac{\tilde{s}}{n^{(S)}} - \frac{\tilde{r}}{n^{(R)}}}{\hat{w}^-} \middle| q_c^S = \frac{\tilde{s}}{n^{(S)}}\right) \\ \times P\left(q_c^S = \frac{\tilde{s}}{n^{(S)}}\right) \end{array} \right] \end{aligned} \tag{3.13}$$

where $n^{(R)} = (1 - \hat{w}^+)n^T$ and $n^{(S)} = \frac{n^{(R)}}{1-\hat{w}^-}$ represent the sizes of hypothetical samples from the remaining and source distributions relative to target sample size n^T. To avoid division by zero, note that $P(q_c^T) = P(q_c^R)$ when $\hat{w}^+ = 0$ and $P(q_c^R) = P(q_c^S)$ when $\hat{w}^- = 0$. Intuitively, this gives $P(q_c^T) = P(q_c^S)$ when both $\hat{w}^+ = 0$ and $\hat{w}^- = 0$. Division by zero in the definition of $n^{(S)}$ is also avoided by treating any case where $\hat{w}^- = 1$ as $(\hat{w}^- = 0$ and $\hat{w}^+ = 1)$ since both cases imply there is no common remaining distribution between the source and target distributions.

The cost of computing this discrete distribution can be decomposed into the costs of computing $P\left(q_c^T = \frac{\tilde{t}}{n^T}\right)\ \forall\ \tilde{t} \in \{0, \ldots, n^T\}$ and $P\left(q_c^R = \frac{\tilde{r}}{n^{(R)}}\right)\ \forall\ \tilde{r} \in \{0, \ldots, n^{(R)}\}$, as the latter can be pre-computed once for use in computing the former. As $n^{(R)}$ is proportional to n^T, the cost of computing all $P(q_c^T)$ is $O(n^T n^{(R)}) = O((n^T)^2)$.

Furthermore, $\max(n^{(S)}) = \frac{\max(n^{(R)})}{\min(1-\hat{w}^-)} = \frac{n^T}{1/n^T} = (n^T)^2$, as $n^{(R)} = n^T$ when $\hat{w}^+ = 0$ and $\hat{w}^-$ must represent a discrete proportion of n^T instances such that $1 - \hat{w}^-$ has a non-zero minimum when $\hat{w}^-$ represents one fewer than all n^T instances ($\hat{w}^- = \frac{n^T-1}{n^T}$). Therefore, the worst-case cost of computing all $P(q_c^R)$ is $O(n^{(R)}n^{(S)}) = O((n^T)^3)$. By reasonably limiting the summation granularities (i.e. limiting maximum $n^{(S)}$ and $n^{(R)}$), both costs can be reduced to $O(n^T)$.

From discrete probability distribution $P(q_c^T)$, the expected value[5] can serve as GSLS quantification estimate $\hat{q}_c^{T\{GSLS\}}$, and an equal-tailed credible interval $[l, u]$ for desired probability γ can serve as a prediction interval:

$$\hat{q}_c^{T\{GSLS\}} = E[P(q_c^T)] \quad \Big| \quad \gamma = P(l \leq q_c^T \leq u) \tag{3.14}$$

3.4 Dynamic Quantifier Selection

The previous section presented GSLS as a method for reliable quantification under more general dataset shift than alternative methods. However, if PCC's assumption of no shift or EM's assumption of prior shift hold, these methods would be expected to provide greater quantification accuracy and narrower prediction intervals. This section presents a method for identifying the most likely condition of shift between given source and target samples to enable the dynamic selection of the most appropriate quantification method. The proposed approach for selecting a quantification method is presented as a decision tree in Fig. 3.3. This decision tree relies on statistical hypothesis tests for two different conditions of shift: a test for whether *any shift at all* is likely (a null hypothesis of $P^S(X) = P^T(X)$) and a test for whether *non-prior shift* is likely (a null hypothesis of $P^S(X|Y) = P^T(X|Y)$). Each test is to be performed between the target

[5] An earlier formulation (Denham et al., 2021) used a maximum-likelihood quantification estimate, but the expected value should provide better accuracy on average in the case of a skewed distribution.

if *any shift at all* is likely **then**
  if the shift is likely to be *non-prior shift* **then**
    Only assume **general shift** and apply GSLS
  else
    Assume **prior shift** and apply EM
  end if
else
  Assume **no shift** and apply PCC
end if

Figure 3.3: Decision tree for selecting a quantification method.

sample and a held-out portion of the source sample not used for training. Rejecting the null hypothesis for either test at some target α is considered a positive detection of the respective shift condition. The following subsections review options for both tests by drawing on existing shift tests from the literature as well as a novel proposal for an efficient non-prior shift test. These options are experimentally evaluated in Section 3.6 to determine the best methodology for selecting an appropriate quantification method.

3.4.1 Testing for Any Dataset Shift

Statistical tests have previously been proposed for detecting the presence of any dataset shift between source and target samples. One approach is to perform a two-sample Kolmogorov-Smirnov (KS) test on classifier outputs $g_c(x)$ for each sample's instances (Lipton, Wang & Smola, 2018; Rabanser et al., 2019). For multi-class tasks with multiple classifier outputs per instance, separate KS tests can be performed on the outputs for each class; a Bonferroni correction can be applied to reduce the test α conservatively, allowing the tests to be treated as independent such that shift is detected if the null hypothesis is rejected for any class. Such a Bonferroni correction has been used successfully to detect shift based on tests of multiple input features (Rabanser et al., 2019).

Additionally, the original authors of the EM method (Saerens et al., 2002) propose a likelihood ratio (LR) test to evaluate whether EM's prior-shift correction will make a significant improvement, which is expected if shift is present. While this method was designed under the assumption that the only possible shift is prior shift, it natively supports multi-class tasks and is particularly suited to detecting shift that can be accounted for by EM. Therefore, it is included in the experimental evaluation.

3.4.2 Testing for Non-Prior Dataset Shift

Two similar tests have recently been proposed for detecting dataset shift while excluding shifts for which the prior-shift assumption holds (A. Maletzke et al., 2018; Vaz et al., 2019). Both tests involve measuring distances between classifier output distributions of target samples and of best-fit mixtures of class-conditional source distributions. Such distances are measured for both the observed target sample and synthetically prior-shifted target samples derived from the source sample. The distribution of distances for prior-shifted samples is then used to calculate the p-value of the distance for the observed target sample. If the observed target distance is greater than the distances for $1 - \alpha$ of the prior-shifted samples, then the null hypothesis is rejected, indicating non-prior shift. Besides implementation details, the two tests differ in the following two fundamental ways.

The tests firstly differ in how they create prior-shifted samples. The weak prior-shift assumption (WPA) test (Vaz et al., 2019) generates samples with the single best-fit class distribution for the observed target sample. In contrast, the concept distance threshold (CDT) test (A. Maletzke et al., 2018) uses a grid of varied class proportions. CDT is more efficient because the samples can be pre-computed for all tests with a given source sample, whereas WPA's samples depend on the observed target sample's estimated class proportions. While producing an exhaustive grid of class proportions may be

impractical for multi-class tasks, this can easily be addressed by randomly sampling class proportions from a symmetric Dirichlet distribution.

The tests also differ in how they measure distances between distributions: the WPA test uses the KS distance, while the CDT test uses the Hellinger Distance (HD). While the original proposal of the WPA test was to use kernel smoothers to estimate class-conditional distributions, the evaluated implementation produces mixture samples via class-conditional re-sampling and estimates distributions with histograms to provide more consistent behaviour on even small sample sizes. As the KS distance is insensitive to having too many bins, it is evaluated using a reasonably large number of 1,000 equal-spaced bins. On the other hand, as HD is very sensitive to the number of histogram bins used, it is evaluated using the same rule-of-thumb for constructing equal-density bins as was used to fit GSLS in Section 3.3.1. Compared to previously proposed binning schemes, this is much more efficient than computing a median distance over varied bin counts (González-Castro, Alaiz-Rodríguez & Alegre, 2013), and initial experiments found it to be more effective than the fixed number of 11 bins proposed by Moreira dos Reis et al. (2018). In order to apply these distance measures to multi-class datasets, the maximum distance is taken over the classifier output distributions for all classes, though similar results in initial experiments were also achieved using a mean distance.

The experiments in Section 3.6 evaluate all four possible combinations of the WPA and CDT sampling methods with the KS and HD distances. P-values are calculated based on 1,000 prior-shifted samples, and the efficiency of both tests is improved by determining the best-fit mixture of class-conditional source distributions through EM quantification instead of minimising the distance over a grid of class proportions.

Finally, a simpler test for detecting non-prior shift is proposed by this research: Adjusted-KS (AKS). AKS simply involves performing a histogram-based two-sample KS test between the classifier outputs of the target sample and the best-fit mixture of source class-conditional distributions, with best-fit class proportions estimated by EM.

If shift is detected between these two distributions, then EM has not been effective at accounting for the shift, and therefore the shift should be considered non-prior shift. As with the KS test for any dataset shift, a Bonferroni correction can be used for multi-class tasks. Note that this proposed method breaks the KS test's assumption of independence between the two samples, as best-fit class proportions are derived from the target sample. However, empirical testing of such a method is worthwhile, given that it avoids the repeated, expensive distance calculations over many simulated samples that are required for the WPA and CDT tests.

3.5 Constrained Quantification Intervals

Producing a quantification prediction interval that is reliable may not be enough to fully meet users' needs if the interval is wider than their desired level of certainty. For example, an interval between 0% and 100% is guaranteed to contain the true proportion of instances belonging to a class, but it provides no information about what that proportion's true value is. Unacceptably wide prediction intervals can be caused by uncertainty in the classifications of individual instances in the target set and uncertainty due to observed dataset shift between the source and target distributions.

This section presents a framework for constraining quantification interval widths to user-specified limits. The framework is based on the premise that all uncertainty in the classification of an individual target instance x_i^T can be eliminated by requesting a class label from the user, which is assumed to be correct: $P(\text{user}(y_i^T) = y_i^T) = 1$. Therefore, as class labels are requested for more instances, the uncertainty in quantifying the whole target set is also reduced. In the extreme case, requesting class labels for the entire target set would eliminate all uncertainty and provide perfectly certain quantification predictions (i.e. intervals with zero width). In order to minimise the manual labelling effort required by users, the proposed framework will request class labels for minimal

subsets of target instances while aiming to ensure that quantification prediction intervals can be produced within user-specified limits.

For example, given a target set of n^T instances, a quantification method may initially produce an 80% prediction interval for a class c that ranges from 0.3 to 0.5 (i.e. $P(0.3 \leq q_c^T \leq 0.5) = 0.8$), having width $0.5 - 0.3 = 0.2$. If the user specifies a maximum interval width of 0.1, then some subset of the n^T target instances should be identified for the user to label manually. By accounting for the user-provided class labels, it should be possible to provide a prediction interval no wider than the specified limit, such as $P(0.36 \leq q_c^T \leq 0.46) = 0.8$ with a width of 0.1.

In practice, the target subsets to be labelled by the user will comprise instances with the greatest combined impact on quantification uncertainty. Requesting labels for such instances bears similarity to the task of *classification with rejection* (aka *selective classification*), where classifiers are designed to choose not to classify (i.e. *reject*) instances that they cannot confidently classify, commonly based on classifier probabilities or other measures of classification uncertainty (Chow, 1970; Ni, Charoenphakdee, Honda & Sugiyama, 2019). While past research has sought to identify instances contributing to dataset shift (Gu, Zhang, Lu & Lin, 2016) and applied requested instance labels to improve quantification (A. G. Maletzke, dos Reis & Batista, 2018), this is the first known work to employ rejection to constrain the uncertainty of quantification estimates within user-specified limits. For consistency with rejection literature, the subset of instances to be labelled by the user is referred to as the *rejected set*, and its complement is referred to as the *selected set*.

3.5.1 Framework for Constrained Intervals

This section presents a general framework for constraining prediction intervals that will be applied to the intervals produced by PCC, EM, and GSLS quantification in

subsections. The problem can be formally defined as performing m quantifications on a target set of n^T instances $\{X^T, Y^T\}^{n^T}$, where each quantification $q^T_{(j)}$ is for a given class $c_j \in Y$ and a subset of target instances $\{X^j, Y^j\} \subseteq \{X^T, Y^T\}$ such that:

$$q^T_{(j)} = \frac{1}{n^j} \sum_{i=1}^{n^j} I\left(y^j_i = c_j\right) \text{ where: } n^j = |\{X^j, Y^j\}| \tag{3.15}$$

This generalises the typical quantification scenario of Equation (3.1) that only considers a single quantification for each class covering all target instances, such that $m = k$ and $\{X^j, Y^j\} = \{X^T, Y^T\}\ \forall\ j$. This generalisation supports simultaneously constraining intervals with globally minimal rejection across quantifications that may have overlapping target sets, such as target sets from a sliding window.

The user is expected to specify a width limit $\bar{\omega}_j$ for each quantification's equal-tailed prediction interval at a fixed probability γ_j, establishing a goal of achieving distributions $P(q^T_{(j)})\ \forall\ j$ that satisfy the following constraint:

$$P(l \leq q^T_{(j)} \leq u) = \gamma_j \text{ where } u - l \leq \bar{\omega}_j \tag{3.16}$$

This constraint can be simplified with an approximate distribution model $\hat{P}(q^T_{(j)})$ having variance σ^2_j. Because it is only required for determining interval widths, only the distribution's shape needs to be parameterised; its location can be ignored. $\hat{P}(q^T_{(j)})$ is used to translate each width limit $\bar{\omega}_j$ into a variance limit $\bar{\sigma}^2_j$:

$$\hat{P}_{\bar{\sigma}^2_j}(l \leq q^T_{(j)} \leq u) = \gamma_j \text{ where } u - l = \bar{\omega}_j \tag{3.17}$$

An appropriate model for $\hat{P}(q^T_{(j)})$ will depend on the quantification method. As long as the shape of $\hat{P}(q^T_{(j)})$ can be parameterised solely by its variance, then an exponential search or similar strategy can be used to find $\bar{\sigma}^2_j$ for a given $\bar{\omega}_j$.

An optimisation problem can now be formulated to constrain the variance of each quantification with minimal rejection. This will optimise a Boolean vector δ that identifies a target subset $\{X^{\delta}, Y^{\delta}\}$ to select (i.e. not reject), such that $\{X^{j,\delta}, Y^{j,\delta}\}$ can be defined as the set of $n^{j,\delta} = |\{X^{j,\delta}, Y^{j,\delta}\}|$ selected instances involved in each quantification j:

$$\begin{aligned} \{X^{\delta}, Y^{\delta}\} &= \{x_i^T, y_i^T \ \forall \ 1 \leq i \leq n^T, \delta_i = 1\} \\ \{X^{j,\delta}, Y^{j,\delta}\} &= \{X^j, Y^j\} \cap \{X^{\delta}, Y^{\delta}\} \end{aligned} \tag{3.18}$$

Because selected instances are the only source of uncertainty (as established at the beginning of this section), each quantification's post-rejection variance $V_{\delta}(q_{(j)}^T)$ can be expressed as the variance of quantifying only selected instances scaled to the total number of quantified instances:

$$V_{\delta}(q_{(j)}^T) = (n^{j,\delta}/n^j)^2 V(q_{(j,\delta)}^T) \tag{3.19}$$

Therefore, the optimisation problem will seek to maximise the number of selected instances (minimising rejection) while ensuring this variance does not exceed the variance limit for each of the m quantifications:

$$\begin{aligned} &\text{maximise } \textstyle\sum_{i=1}^{n^T} \delta_i \\ &\text{subject to } (n^{j,\delta}/n^j)^2 \hat{V}(q_{(j,\delta)}^T) \leq \bar{\sigma}_j^2 \ \ \forall \ \ 1 \leq j \leq m \\ &\text{and } \delta \in \{0,1\}^{n^T} \end{aligned} \tag{3.20}$$

where the definition of $\hat{V}(q_{(j,\delta)}^T) \approx V(q_{(j,\delta)}^T)$ depends on the quantification method in use. As this problem optimises Boolean variables in δ to maximise an objective while subject to constraints, it is a binary (or 0-1) integer programming (BIP) problem (Williams, 2009). Given that the maximisation objective is linear (and therefore convex), if $(n^{j,\delta}/n^j)^2 \hat{V}(q_{(j,\delta)}^T)$ in each constraint is a convex function of δ, then it is a convex

BIP problem (Boyd, Boyd & Vandenberghe, 2004) that can be solved with standard software (Agrawal, Verschueren, Diamond & Boyd, 2018).

The following subsections identify an appropriate distribution model $\hat{P}(q^T_{(j)})$ and post-selection variance function $\hat{V}(q^T_{(j,\delta)})$ for the PCC, EM, and GSLS quantification methods. $\hat{V}(q^T_{(j,\delta)})$ may be defined in subsequent sections as only an approximation of the true variance to ensure the BIP problem constraints are convex.

3.5.2 Constrained PCC Intervals

In order to complete the constrained intervals problem for the PCC quantification method, $\hat{P}^{\text{PCC}}(q^T_{(j)})$ and $\hat{V}^{\text{PCC}}(q^T_{(j,\delta)})$ must be defined to reflect PCC's scaled Poisson binomial (PB) distribution. While a PB distribution's shape cannot be parameterised solely by its variance, it can be approximated with a Gaussian distribution. Therefore, the variance limits can be determined from interval width limits for PCC with $\hat{P}^{\text{PCC}}(q^T_{(j)}) = \mathcal{N}(0, \sigma^2_j)$. $\hat{V}^{\text{PCC}}(q^T_{(j,\delta)})$ can easily be defined based on the definition of PB variance (Keith & O'Connor, 2018): $\sum_{i=1}^{n} p_i(1-p_i)$ where $p \in (0,1)^n$ represents n event probabilities. Combining this with the earlier definition of PCC quantification in Equation (3.3) gives:

$$\hat{V}^{\text{PCC}}(q^T_{j,\delta}) = \frac{1}{n^{j,\delta 2}} \sum_{i=1}^{n^{j,\delta}} g_{c_j}(x_i^{j,\delta})(1 - g_{c_j}(x_i^{j,\delta})) \tag{3.21}$$

where the scaling of the PB mean in Equation (3.3) is mirrored by scaling the variance by the inverse square of the number of selected instances $n^{j,\delta}$. After cancelling out $n^{j,\delta}$ terms, constraints on $(n^{j,\delta}/n^j)^2\hat{V}^{\text{PCC}}(q^T_{(j,\delta)})$ will be convex because it is a linear function of δ.

3.5.3 Constrained EM Intervals

The $\hat{P}^{\text{EM}}(q^T_{(j)})$ and $\hat{V}^{\text{EM}}(q^T_{(j,\delta)})$ needed to constrain EM intervals can be defined by leveraging the fact that EM intervals are actually confidence intervals derived from EM's maximum-likelihood estimate of $q^T_{(j)}$ (Saerens et al., 2002; Tasche, 2019). Because the quantification estimate's distribution is asymptotically normal (Tasche, 2019), a Gaussian distribution can again be used to model $\hat{P}^{\text{EM}}(q^T_{(j)})$. The variance of this distribution is the reciprocal of the Fisher information $\mathcal{I}(q^T_{(j,\delta)})$, which is itself the negative curvature of the marginal log-likelihood of observed data $X^{j,\delta}$ over $q^T_{(j,\delta)}$ (Tasche, 2019; Keith & O'Connor, 2018):

$$\begin{aligned}\mathcal{I}(q^T_{(j,\delta)}) &= -E\left[\frac{\partial^2}{\partial {q^T_{(j,\delta)}}^2}\ln P(X^{j,\delta})\middle| q^T_{(j,\delta)}\right]\\ &= -\frac{\partial^2}{\partial {q^T_{(j,\delta)}}^2}\left(\ln\prod_{i=1}^{n^{j,\delta}} P(x_i^{j,\delta}|q^T_{(j,\delta)})\right)\\ &= \sum_{i=1}^{n^{j,\delta}}\frac{\partial^2}{\partial {q^T_{(j,\delta)}}^2}\left(-\ln P(x_i^{j,\delta}|q^T_{(j,\delta)})\right)\end{aligned} \tag{3.22}$$

Therefore, the quantification variance can be approximated as:

$$\hat{V}^{\text{EM}}(q^T_{(j,\delta)}) = \left(\sum_{i=1}^{n^{j,\delta}}\frac{\partial^2}{\partial {q^T_{(j)}}^2}\left(-\ln P(x_i^{j,\delta}|\hat{q}^T_{(j)})\right)\right)^{-1} \tag{3.23}$$

where estimate $\hat{q}^T_{(j)}$ over all quantified instances is used as an approximation for $q^T_{(j,\delta)}$ over the selected subset because it is the expected value over all possible δ (i.e. $E[q^T_{(j,\delta)}] = q^T_{(j)}$) and because varying the value with δ would no longer guarantee that $(n^{j,\delta}/n^j)^2\hat{V}^{\text{EM}}(q^T_{(j,\delta)})$ is convex. Each instance's $\frac{\partial^2}{\partial {q^T_{(j)}}^2}\left(-\ln P(x_i^{j,\delta}|q^T_{(j)})\right)$ can be

calculated from the EM-estimated $\hat{q}^T_{(j)}$, classification probability $g_{c_j}(x_i^{j,\delta})$, and estimated source class prior $\hat{P}_S(c_j)$:

$$
\begin{aligned}
P(x_i|q^T_{(j)}) &= P(x_i, y_i = c_j|q^T_{(j)}) + P(x_i, y_i \neq c_j|q^T_{(j)}) \\
&= q^T_{(j)} P_S(x_i|y_i = c_j) + (1 - q^T_{(j)}) P_S(x_i|y_i \neq c_j) \\
\therefore \frac{\partial^2}{\partial {q^T_{(j)}}^2}\left(-\ln P(x_i|q^T_{(j)})\right) &= \left(\frac{P_S(x_i|y_i = c_j) - P_S(x_i|y_i \neq c_j)}{q^T_{(j)} P_S(x_i|y_i = c_j) + (1 - q^T_{(j)}) P_S(x_i|y_i \neq c_j)}\right)^2 \\
&= \left(\frac{\frac{P_S(y_i=c_j|x_i)P_S(x_i)}{P_S(y_i=c_j)} - \frac{P_S(y_i\neq c_j|x_i)P_S(x_i)}{P_S(y_i\neq c_j)}}{\frac{q^T_{(j)}P_S(y_i=c_j|x_i)P_S(x_i)}{P_S(y_i=c_j)} + \frac{(1-q^T_{(j)})P_S(y_i\neq c_j|x_i)P_S(x_i)}{P_S(y_i\neq c_j)}}\right)^2 \\
&= \left(\frac{\frac{P_S(y_i=c_j|x_i)}{P_S(y_i=c_j)} - \frac{P_S(y_i\neq c_j|x_i)}{P_S(y_i\neq c_j)}}{\frac{q^T_{(j)}P_S(y_i=c_j|x_i)}{P_S(y_i=c_j)} + \frac{(1-q^T_{(j)})P_S(y_i\neq c_j|x_i)}{P_S(y_i\neq c_j)}}\right)^2 \\
&\approx \left(\frac{\frac{g_{c_j}(x_i)}{\hat{P}_S(c_j)} - \frac{(1-g_{c_j}(x_i))}{(1-\hat{P}_S(c_j))}}{\frac{\hat{q}^T_{(j)}g_{c_j}(x_i)}{\hat{P}_S(c_j)} + \frac{(1-\hat{q}^T_{(j)})(1-g_{c_j}(x_i))}{(1-\hat{P}_S(c_j))}}\right)^2
\end{aligned}
\tag{3.24}
$$

The convexity requirement for constraints on $(n^{j,\delta}/n^j)^2\hat{V}^{\text{EM}}(q^T_{(j,\delta)})$ is satisfied because they can be expressed with second-order cones (Boyd et al., 2004).

3.5.4 Constrained GSLS Intervals

The $\hat{P}^{\text{GSLS}}(q^T_{(j)})$ and $\hat{V}^{\text{GSLS}}(q^T_{(j,\delta)})$ needed to constrain GSLS intervals must be defined based on the quantification distribution of the GSLS model, which can be fully formulated from the partial distributions in Equations (3.8) and (3.9):

$$
\begin{aligned}
P^{\text{GSLS}}(q^T) &= P\left(w^+q^+ + (1 - w^+)q^R\right) \\
&= P\left(w^+q^+ + \frac{1 - w^+}{1 - w^-}\left(q^S - w^-q^-\right)\right)
\end{aligned}
\tag{3.25}
$$

Because $P^{\text{GSLS}}(q^T)$ is a mixture of PB distribution $P(q^S)$ and uniform distributions $P(q^+)$ and $P(q^-|q^S)$, $\hat{P}^{\text{GSLS}}(q^T_{(j)})$ can be conservatively approximated as a uniform

distribution with variance $\bar{\sigma}_j^2$, such that the intervals it produces should be at least as wide as fully modelled intervals. Applying the rules for variance of sums and differences of random variables and accounting for the fact that $P(q^+)$ is independent of $P(q^S)$ and $P(q^-)$, the variance of GSLS can be defined in terms of the variances of its component distributions:

$$\begin{aligned} V^{\text{GSLS}}\left(q^T\right) &= V\left(w^+q^+ + \frac{1-w^+}{1-w^-}\left(q^S - w^-q^-\right)\right) \\ &= V(w^+q^+) + \left(\frac{1-w^+}{1-w^-}\right)^2\left(V(q^S) + V\left(w^-q^-\right) - 2\text{cov}(q^S, w^-q^-)\right) \\ &= {w^+}^2V\left(q^+\right) + \left(\frac{1-w^+}{1-w^-}\right)^2\left(V(q^S) + w^{-2}V(q^-) - 2w^-\text{cov}(q^S, q^-)\right) \end{aligned} \tag{3.26}$$

This formulation can be further decomposed and applied to express $V^{\text{GSLS}}\left(q^T_{(j,\delta)}\right)$ as a sum of three terms:

$$\begin{aligned} V^{\text{GSLS}}\left(q^T_{(j,\delta)}\right) &= \psi^+_{(j,\delta)} + \psi^S_{(j,\delta)} + \psi^-_{(j,\delta)} \\ \psi^+_{(j,\delta)} &= {w^+_{(j,\delta)}}^2 V\left(q^+_{(j,\delta)}\right) \\ \psi^S_{(j,\delta)} &= \left(\frac{1-w^+_{(j,\delta)}}{1-w^-_{(j,\delta)}}\right)^2 V\left(q^S_{(j,\delta)}\right) \\ \psi^-_{(j,\delta)} &= \left(\frac{w^-_{(j,\delta)}\left(1-w^+_{(j,\delta)}\right)}{1-w^-_{(j,\delta)}}\right)^2 \epsilon_{(j,\delta)} \\ \epsilon_{(j,\delta)} &= V\left(q^-_{(j,\delta)}\right) - \frac{2}{w^-_{(j,\delta)}}\text{cov}\left(q^S_{(j,\delta)}, q^-_{(j,\delta)}\right) \end{aligned} \tag{3.27}$$

where $w^+_{(j,\delta)}$ and $w^-_{(j,\delta)}$ represent the effective gain and loss weights for selected target subset $\{X^{j,\delta}, Y^{j,\delta}\}$. Because GSLS fits b-bin histogram h^+ to model the distribution of gain across target distribution histogram h^T (Section 3.3.1), $\hat{w}^+_{(j,\delta)}$ can be calculated as

the mean of instance-specific weights $\hat{w}_i^+$ that are proportional to the ratio of h^+ and h^T bins containing instance $x_i^{j,\delta}$:

$$\hat{w}_{(j,\delta)}^+ = \frac{1}{n^{j,\delta}} \sum_{i=1}^{n^{j,\delta}} \hat{w}_i^+ \quad \Big| \quad \hat{w}_i^+ = \frac{h^+(x_i^{j,\delta})}{h^T(x_i^{j,\delta})} \hat{w}_{(j)}^+ \tag{3.28}$$

where $h^+(x_i^{j,\delta})$ and $h^T(x_i^{j,\delta})$ are the histogram bins containing instance $x_i^{j,\delta}$, and $\hat{w}_i^+$ indexes weights of subset $\{X^{j,\delta}, Y^{j,\delta}\}$ for notational brevity. For example, this can be applied to recover the original $\hat{w}^+$ from the full target sample of n^T instances:

$$\begin{aligned} \frac{1}{n^T} \sum_{i=1}^{n^T} \hat{w}_i^+ &= \frac{1}{n^T} \sum_{i=1}^{n^T} \frac{h_i^+(x_i^T)}{h^T(x_i^T)} \hat{w}^+ \\ &= \frac{1}{n^T} \sum_{i=1}^{b} n^T h_i^T \frac{h_i^+}{h_i^T} \hat{w}^+ \\ &= \hat{w}^+ \sum_{i=1}^{b} h_i^+ \\ &= \hat{w}^+ \end{aligned} \tag{3.29}$$

The effective loss weight cannot be determined similarly, as there is no direct relationship between the distribution of loss across the source distribution and the distribution of target instances. Therefore, $w_{(j,\delta)}^-$ is approximated by the pre-selection loss weight $\hat{w}_{(j)}^-$, which is the expected value over all possible δ.

The variance terms $\psi_{(j,\delta)}^+$, $\psi_{(j,\delta)}^S$, and $\psi_{(j,\delta)}^-$ can be approximately calculated to formulate a $\hat{V}^{\text{GSLS}}(q_{(j,\delta)}^T)$ that meets the convexity requirement. The first term $\psi_{(j,\delta)}^+$ is the product of the variance of 0-1 uniform distribution $P(q_{(j,\delta)}^+)$ and gain weight $w_{(j,\delta)}^+$, which is approximated as the mean of selected $\hat{w}_i^+$ values:

$$\begin{aligned} \psi_{(j,\delta)}^+ &\approx V\left(q_{(j,\delta)}^+\right) \left(\frac{1}{n^{j,\delta}} \sum_{i=1}^{n^j} \hat{w}_i^+ \right)^2 \\ &= \frac{1}{{n^{j,\delta}}^2} \left(\sum_{i=1}^{n^{j,\delta}} \hat{w}_i^+ \sqrt{V\left(q_{(j,\delta)}^+\right)} \right)^2 \end{aligned} \tag{3.30}$$

The second term $\psi^S_{(j,\delta)}$ can use the same formulation as PCC in Equation (3.21) for the variance of source distribution $P(q^S_{(j,\delta)})$. While $\hat{w}^+_{(j,\delta)}$ cannot be computed as a mean of w^+_i weights without breaking the guarantee of convexity, each instance's contribution to the source variance can still be weighted by its local weight w^+_i:

$$\begin{aligned}\psi^S_{(j,\delta)} &\approx \left(\frac{1-\hat{w}^+_{(j,\delta)}}{1-\hat{w}^-_{(j,\delta)}}\right)^2 \frac{1}{n^{j,\delta 2}} \sum_{i=1}^{n^{j,\delta}} g_{c_j}(x_i^{j,\delta})(1-g_{c_j}(x_i^{j,\delta})) \\ &\approx \frac{1}{n^{j,\delta 2}} \sum_{i=1}^{n^{j,\delta}} \left(\frac{1-\hat{w}^+_i}{1-\hat{w}^-_{(j)}}\right)^2 g_{c_j}(x_i^{j,\delta})(1-g_{c_j}(x_i^{j,\delta}))\end{aligned} \tag{3.31}$$

The third and final component $\psi^-_{(j,\delta)}$ can use the same approach to approximating gain and loss weights as in Equation (3.31). Additionally, $\epsilon_{(j,\delta)}$ must be non-negative to guarantee convexity, and this is ensured through the use of a $\max$ operation:

$$\begin{aligned}\psi^-_{(j,\delta)} &\approx \frac{1}{n^{j,\delta 2}} \left(\sum_{i=1}^{n^{j,\delta}} \frac{\hat{w}^-_{(j)}(1-\hat{w}^+_i)}{1-\hat{w}^-_{(j)}}\right)^2 \hat{\epsilon}_{(j,\delta)} \\ &\qquad \text{where: } \hat{\epsilon}_{(j,\delta)} = V\Big(q^-_{(j,\delta)}\Big) - \frac{2}{\hat{w}^-_{(j)}} \text{cov}\Big(q^S_{(j,\delta)}, q^-_{(j,\delta)}\Big) \\ &\leq \frac{1}{n^{j,\delta 2}} \left(\sum_{i=1}^{n^{j,\delta}} \frac{\hat{w}^-_{(j)}(1-\hat{w}^+_i)}{1-\hat{w}^-_{(j)}} \sqrt{\max(0, \hat{\epsilon}_{(j,\delta)})}\right)^2\end{aligned} \tag{3.32}$$

Note that this $\max$ operation will conservatively over-estimate the variance, and only in cases where $V(q^S) > V(q^S - w^- q^-)$. The $V\Big(q^-_{(j,\delta)}\Big)$ and $\text{cov}\Big(q^S_{(j,\delta)}, q^-_{(j,\delta)}\Big)$ terms are treated as constants approximated from expectations of distributions $P(q^S_{(j,\delta)})$ and $P(q^-_{(j,\delta)}|q^S_{(j,\delta)})$ from Section 3.3.3:

$$
\begin{aligned}
V(q^-) &= E\Big[V(q^-|q^S)\Big] + V\Big(E[q^-|q^S]\Big) \\
&\quad \text{(by the law of total variance)} \\
&= E\Big[V(q^-|q^S)\Big] + E\Big[E[q^-|q^S]^2\Big] - E\Big[E[q^-|q^S]\Big]^2 \\
&\quad \text{(by the definition of variance)} \\
&= E\Big[V(q^-|q^S) + E[q^-|q^S]^2\Big] - E\Big[E[q^-|q^S]\Big]^2 \\
&\quad \text{(by the linearity of expectation)} \\
\operatorname{cov}(q^S, q^-) &= E[q^S q^-] - E[q^S]E[q^-] \\
&\quad \text{(by the definition of covariance)} \\
&= E\Big[E[q^S q^-|q^S]\Big] - E[q^S]E\Big[E[q^-|q^S]\Big] \\
&\quad \text{(by the law of total expectation)} \\
&= E\Big[q^S E[q^-|q^S]\Big] - E[q^S]E\Big[E[q^-|q^S]\Big] \\
&\quad \text{(by the linearity of expectation)}
\end{aligned}
\tag{3.33}
$$

where $V(q^-|q^S)$ and $E[q^-|q^S]$ are the variance and mean of uniform distribution $P(q^-|q^S)$ and the remaining expectations can be computed numerically by approximating $P(q^S)$ with a Gaussian distribution.

Finally, Equations (3.30), (3.31), and (3.32) can be combined into a full approximation for the GSLS variance:

$$
\begin{aligned}
\hat{V}^{\text{GSLS}}\big(q^T_{(j,\delta)}\big) &= \hat{\psi}^+_{(j,\delta)} + \hat{\psi}^S_{(j,\delta)} + \hat{\psi}^-_{(j,\delta)} \\
\hat{\psi}^+_{(j,\delta)} &= \frac{1}{n^{j,\delta 2}} \left(\sum_{i=1}^{n^{j,\delta}} \hat{w}^+_i \sqrt{V\Big(q^+_{(j,\delta)}\Big)} \right)^2 \\
\hat{\psi}^S_{(j,\delta)} &= \frac{1}{n^{j,\delta 2}} \sum_{i=1}^{n^{j,\delta}} \left(\frac{1-\hat{w}^+_i}{1-\hat{w}^-_{(j,\delta)}} \right)^2 g_{c_j}(x_i^{j,\delta})(1-g_{c_j}(x_i^{j,\delta})) \\
\hat{\psi}^-_{(j,\delta)} &= \frac{1}{n^{j,\delta 2}} \left(\sum_{i=1}^{n^{j,\delta}} \frac{\hat{w}^-_{(j,\delta)}(1-\hat{w}^+_i)}{1-\hat{w}^-_{(j,\delta)}} \sqrt{\max(0, \hat{\epsilon}_{j,\delta})} \right)^2
\end{aligned}
\tag{3.34}
$$

After cancelling out $n^{j,\delta}$ terms, constraints on $(n^{j,\delta}/n^j)^2\hat{V}^{\text{GSLS}}(q^T_{(j,\delta)})$ meet the convexity requirement as they can be expressed with second-order cones.

While several approximations have been necessary to apply the optimisation problem to each quantification method, experimental results in Section 3.6 demonstrate that these approximations are adequate for the purpose of constraining prediction intervals.

3.6 Experimental Study

The following experiments evaluate the GSLS model, the decision tree for dynamic quantification method selection, and the framework for constraining quantification intervals. The GSLS model is compared to the PCC and EM quantification methods, focusing on prediction interval quality. The performance of the GSLS fitting method is further analysed to understand when shift misestimation may negatively impact prediction intervals, and the sensitivity of GSLS to the choice of histogram bins b is also evaluated. The shift tests identified in Section 3.4 are compared to determine the best tests for dynamically selecting a quantification method, and the framework for constraining quantification intervals proposed in Section 3.5 is compared to baselines that constrain intervals with confidence-based rejection. Finally, experiments are performed using synthetic datasets with known distributions to further demonstrate the effectiveness of the methods for producing and constraining quantification intervals without the confounding factor of imperfectly calibrated classification probabilities. All experiments were performed on 64-bit Ubuntu 20.04 running on a 12x2.60GHz Intel Core i7 CPU with 48GB of memory. Source-code for the experiments (including dataset pre-processing) has been made available online at: `https://github.com/ben-denham/gsls`.

3.6.1 Experiment Framework

Evaluating the quantification methods under varied shift conditions required a means to control the degree of shift between source and target distributions. Experiments are performed with benchmark datasets that can be divided into natural "concepts" that are expected to vary in both feature distribution $P(X)$ and classification rule $P(Y|X)$. For each experiment on a dataset, three concepts are randomly selected to represent the gain P^+, loss P^-, and remaining P^R distributions. For specified w^+ and w^- parameters, a source sample is composed of $w^- n^S$ instances from P^- and $((1-w^-)n^S)$ instances from P^R, and a target sample is composed of $w^+ n^T$ instances from P^+ and $((1-w^+)n^T)$ instances from P^R (sampled from P^R instances that were not selected for the source sample). To ensure quantification across concepts was non-trivial, instances were sampled from each concept according to a random class distribution drawn from $\text{Dir}(\{1\}^k)$. This controlled shift will be referred to as *GSLS shift*. Experiments were also performed with *prior shift* by constructing source and target samples from the same randomly selected concept but sampled according to different random class distributions drawn from $\text{Dir}(\{1\}^k)$.

Experiments are performed with shift introduced to benchmark datasets used in previous studies of shift and quantification (A. Maletzke et al., 2018; D. M. dos Reis, Maletzke & Batista, 2018; D. dos Reis et al., 2018; A. Maletzke et al., 2020). These datasets have at least three concepts to represent P^+, P^-, and P^R, and represent a range of feature, class (k), instance (n^S and n^T), and concept counts, as shown in Table 3.1. HLL and HLA contain features of handwritten letters *g*, *p*, and *q* by ten authors. In HLL, the letters are the classes while the authors' writing styles represent concepts, whereas in HLA, the authors are the classes and the letters are the concepts. DIG (sourced from the UCI Machine Learning Repository; Dua & Graff, 2017) contains mean MFCC features of spoken Arabic digits. The sex of the speaker is the class, while the digits

Table 3.1: Benchmark datasets for quantification.

Label	**Description**	**Concepts**	**Features**	k **classes**	n^S	n^T
HLL	Handwritten letters	10	63	3 letters	135	45
HLA	Handwritten letters	3	63	10 authors	135	45
DIG	Arabic digits	10	13	2 sexes	330	110
ISX	Insect wing beats	6	26	2 sexes	1500	500
ISP	Insect wing beats	6	26	2 species	1500	500

being spoken serve as concepts. The features of ISX and ISP are sensor measurements of insect wing beats. The class is the insect sex for ISX and the insect species for ISP. For both datasets, range bands of environmental temperature serve as concepts, as wing beat features vary with temperature (A. Maletzke et al., 2018). Given that temperature is expected to change over time, the different mixtures of temperature ranges between source and target samples for ISX and ISP can be considered representative of drift over the course of a data stream.

Experiments are also performed with variations of a dataset exhibiting real-world shift: a plankton image classification dataset that has been studied in past quantification research (González et al., 2019). The dataset comprises 964 water samples with a range of instance counts taken over nine years, such that dataset shift can be expected to affect the proportions of 51 target plankton classes when attempting to quantify each sample. The engineered features of the prior work (González et al., 2019) are used as their deep features were not provided in a pre-computed form, and their setup is repeated by using the first 200 samples as a source sample but excluding two classes that occur in the source sample less than the two times required for train/calibration splitting. As the original plankton dataset (OPL) has high class imbalance (only six classes have prevalence >1% in the source sample), experiments are also performed with two class set variations: five "functional groups" (FPL) and a binary split (BPL) between the most common class ("mix", ~73% of the source sample) and all other classes.

All quantification methods were applied to probabilistic outputs from scikit-learn's logistic regression classifier, trained for a maximum of 10,000 iterations using the `lbfgs` solver and a multinomial loss for multi-class datasets. Logistic regression was selected for its ability to provide reasonably well-calibrated probabilistic outputs. While a separate calibration step may improve the performance of EM under prior shift (Esuli et al., 2020), this was not performed as it was deemed that the available instances for some datasets would be insufficient to support reliable post-calibration. Half of each source sample was used to train the classifier, while the other half was used as a "calibration set" to construct the source distribution's histogram of classifier outputs: h^S. To present result variations across experiments and not across target classes within each dataset, means and standard deviations are reported over pre-aggregated means across target classes.

3.6.2 Comparison of Quantification Methods

PCC, EM, and GSLS quantification were first compared under GSLS shift (with varied w^+ and w^-) and prior shift (PS). Experiments were performed with 1,000 random source and target samples for each shift condition. For GSLS shift, the evaluated methods also included applying the GSLS model with the true w^+ and w^- used to generate the samples in place of the estimated $\hat{w}^+$ and $\hat{w}^-$ (this variant is referred to as *TGSLS*). The evaluation of interval coverage also includes prediction intervals produced for EM via the bootstrapping method of Tasche (2019) (*EM-BS*), where each bootstrapped sample comprises outputs from the same classifier (without re-training) for computational efficiency.

Table 3.2 presents the *coverage* achieved by 80% prediction intervals of each quantification method over all random samples and target classes for a given dataset and shift condition. *Coverage* is defined as the percentage of prediction intervals that

Table 3.2: Coverage of true class proportions by 80% prediction intervals.

	$\{w^+, w^-\}$	$\{0,0\}$	$\{0,.3\}$	$\{0,.7\}$	$\{0,1\}$	$\{.3,0\}$	$\{.3,.3\}$	$\{.3,.7\}$	$\{.3,1\}$	$\{.7,0\}$	$\{.7,.3\}$	$\{.7,.7\}$	$\{.7,1\}$	$\{1,0\}$	$\{1,.3\}$	$\{1,.7\}$	$\{1,1\}$	**PS**
HLL	PCC	**82%**	**84%**	**82%**	40%	62%	71%	70%	39%	47%	58%	58%	40%	40%	52%	52%	40%	71%
	EM	**100%**	**100%**	**97%**	67%	**89%**	**93%**	**91%**	64%	74%	**84%**	**82%**	63%	64%	77%	76%	66%	**98%**
	EM-BS	**100%**	**98%**	**94%**	65%	**92%**	**94%**	**92%**	66%	78%	**86%**	**84%**	66%	64%	76%	74%	63%	**99%**
	GSLS	**100%**	**99%**	**99%**	**86%**	**96%**	**98%**	**98%**	**92%**	**93%**	**94%**	**94%**	**93%**	**86%**	**89%**	**89%**	**86%**	**96%**
	TGSLS	**81%**	**98%**	**100%**	**83%**	**93%**	**98%**	**91%**	**94%**	**90%**	**93%**	**86%**	**94%**	**84%**	**84%**	**84%**	**84%**	N.A.
HLA	PCC	**81%**	74%	63%	26%	60%	63%	57%	26%	40%	43%	42%	26%	26%	31%	30%	25%	62%
	EM	**95%**	**93%**	80%	67%	**80%**	**83%**	76%	64%	56%	64%	65%	63%	67%	69%	70%	67%	**89%**
	EM-BS	**92%**	76%	53%	21%	75%	68%	53%	23%	48%	48%	40%	22%	20%	23%	23%	20%	**89%**
	GSLS	**99%**	**97%**	**90%**	63%	**92%**	**93%**	**88%**	68%	74%	76%	75%	68%	62%	66%	65%	62%	**88%**
	TGSLS	**81%**	**84%**	**93%**	43%	77%	**92%**	66%	51%	51%	57%	44%	51%	44%	44%	44%	44%	N.A.
DIG	PCC	**96%**	**92%**	72%	21%	45%	53%	48%	19%	28%	33%	32%	18%	21%	26%	25%	20%	49%
	EM	**100%**	**96%**	76%	43%	78%	**85%**	66%	36%	52%	63%	55%	35%	41%	49%	49%	40%	**95%**
	EM-BS	**99%**	**93%**	76%	43%	**83%**	**89%**	76%	41%	58%	71%	62%	41%	40%	50%	52%	40%	**94%**
	GSLS	**99%**	**97%**	**95%**	75%	**94%**	**96%**	**97%**	**87%**	**89%**	**90%**	**92%**	**89%**	77%	77%	78%	76%	**87%**
	TGSLS	**96%**	**99%**	**100%**	80%	**93%**	**98%**	**92%**	**96%**	**93%**	**94%**	**89%**	**96%**	**80%**	**80%**	**80%**	**80%**	N.A.
ISX	PCC	80%	41%	25%	12%	24%	27%	19%	10%	14%	17%	16%	10%	10%	14%	11%	10%	20%
	EM	**99%**	79%	63%	42%	61%	63%	52%	33%	51%	53%	50%	32%	41%	44%	48%	44%	**97%**
	EM-BS	**100%**	**84%**	68%	48%	69%	70%	60%	38%	56%	60%	56%	38%	47%	48%	52%	49%	**98%**
	GSLS	**97%**	**92%**	**82%**	74%	**91%**	**94%**	**91%**	**85%**	**87%**	**87%**	**88%**	**87%**	76%	75%	75%	76%	**84%**
	TGSLS	**80%**	**98%**	**96%**	80%	**92%**	**97%**	**89%**	**96%**	**93%**	**94%**	**87%**	**97%**	**82%**	**82%**	**82%**	**82%**	N.A.
ISP	PCC	74%	19%	8%	4%	16%	16%	9%	4%	8%	9%	8%	4%	5%	6%	6%	5%	7%
	EM	**90%**	20%	11%	8%	14%	18%	10%	6%	6%	9%	9%	5%	8%	8%	8%	9%	74%
	EM-BS	**94%**	22%	10%	8%	16%	22%	13%	8%	8%	11%	10%	6%	7%	6%	6%	8%	**82%**
	GSLS	**96%**	64%	51%	52%	64%	61%	55%	54%	57%	52%	57%	58%	51%	47%	48%	52%	40%
	TGSLS	74%	**83%**	**85%**	80%	**80%**	**87%**	**86%**	**96%**	**87%**	**87%**	**85%**	**97%**	**82%**	**82%**	**82%**	**82%**	N.A.

contain the true class proportion (Tasche, 2019). 80% intervals are used instead of more confident 90% or 95% intervals due to the inherent uncertainty in the task of quantification; intervals with higher confidence may often be too wide to provide users with useful information. Settings where the (unrounded) coverage achieved the intended 80% are noted with bold font. The following can be observed from Table 3.2:

- When no shift has occurred ($w^+ = 0$, $w^- = 0$), all methods achieve coverage above or near 80%.
- Unlike PCC and EM, GSLS achieves coverage above or near 80% under GSLS shift, excepting extreme shifts ($w^+ = 1$ or $w^- = 1$) and the ISP dataset.
- TGSLS generally achieves coverage above or near 80%, indicating that cases of poor GSLS coverage are caused by underestimation of w^+ and w^-. The HLA dataset with ten classes is the exception to this trend. Using a Beta distribution instead of a uniform distribution to model gain and loss class proportions could better fit expected class distributions in such high cardinality datasets (as seen in earlier experimental results; Denham et al., 2021), but this would add complexity to the variance formulations for constraining quantification intervals in Section 3.5.
- Under prior shift, EM achieves the highest coverage (it is designed for this condition), but GSLS achieves coverage above or near 80% for all datasets except ISP.
- EM intervals produced with bootstrapping (EM-BS) have similar or slightly improved coverage compared to native intervals, except for the poorer coverage of EM-BS on the HLA dataset. It is worth highlighting that HLA has more classes than the other datasets, and the bootstrapping method was originally proposed for binary problems (Tasche, 2019).

As PCC and EM intervals have insufficient coverage under many of the analysed shift conditions, evaluation of their widths is irrelevant. The widths of GSLS and T-GSLS intervals are evaluated in Section 3.6.3.

The absolute error (AE) of quantification estimates ($|q_c^T - \hat{q}_c^T|$) was also compared to a baseline achieved by the classify-and-count (CC) method. Table 3.3 presents the mean AE for each method and dataset across experiments with no shift ($w^+ = 0$ and $w^- = 0$), GSLS shift ($w^+ > 0$ or $w^- > 0$), and prior shift. Results that are significantly different to CC according to a corrected re-sampled t-test (Nadeau & Bengio, 1999; $\alpha = 0.05$) are highlighted in bold. PCC has the least AE under no shift, while EM has the least AE under prior shift. This is expected, as these methods are designed to operate most effectively under these types of shift. CC and PCC generally have the least AE under GSLS shift, though substantially higher AE is observed for all methods, highlighting its challenging nature. Except for ISP (for which all methods have very high AE), the AE of GSLS quantification is not less than CC or PCC. Given that GSLS quantification considers a wide range of unlikely "worst-case" target distributions to produce its more reliable intervals, it is not surprising that point estimates by methods that more optimistically assume less drastic shifts are more accurate in practice. As noted earlier, improving accuracy is intractable without any assumptions about the nature of shift (Alexandari et al., 2020).

Table 3.4 reports the coverage and error statistics for quantification on the plankton datasets with real-world shift. As the CC method does not produce quantification intervals, its coverage is computed based on whether point estimates exactly equal true class proportions. Given the high class imbalance in the OPL dataset, separate statistics are reported for quantification targets with less than and greater than 1% prevalence. GSLS achieves much higher coverage than the other quantification methods, except when quantifying low prevalence OPL targets where PCC and even CC point estimates have better coverage than both GSLS and native EM intervals. This suggests

Table 3.3: Mean (with std. dev.) quantification absolute error in percentage points.

		No Shift (1,000 samples)		**GSLS Shift (15,000 samples)**		**Prior Shift (1,000 samples)**	
HLL	CC	1.04	(1.17)	7.36	(10.16)	2.45	(5.23)
	PCC	1.14	(1.02)	7.89	(9.29)	3.23	(5.19)
	EM	1.24	(1.72)	8.58	(11.06)	1.53	(2.50)
	GSLS	**3.34**	**(1.76)**	12.58	(8.85)	8.39	(7.84)
HLA	CC	2.48	(<1)	9.35	(4.14)	4.56	(2.02)
	PCC	2.27	(<1)	**8.20**	**(3.79)**	4.62	(1.75)
	EM	3.21	(1.28)	11.32	(4.29)	4.04	(1.73)
	GSLS	**4.62**	**(1.35)**	12.82	(5.23)	**8.61**	**(3.55)**
DIG	CC	0.55	(<1)	8.52	(12.48)	3.26	(8.60)
	PCC	0.74	(<1)	9.29	(11.79)	5.12	(8.36)
	EM	0.83	(1.25)	10.14	(13.33)	2.08	(2.95)
	GSLS	2.47	(2.18)	11.86	(11.46)	11.09	(12.51)
ISX	CC	0.50	(<1)	8.28	(12.42)	2.76	(5.53)
	PCC	0.47	(<1)	8.84	(11.90)	3.90	(5.87)
	EM	0.48	(<1)	7.72	(12.07)	0.77	(1.59)
	GSLS	1.42	(1.35)	11.67	(12.48)	12.81	(14.63)
ISP	CC	4.49	(3.17)	21.94	(19.76)	20.44	(19.15)
	PCC	1.24	(1.02)	19.64	(17.80)	20.10	(17.08)
	EM	1.80	(1.50)	26.72	(21.47)	3.04	(4.85)
	GSLS	2.01	(1.51)	19.17	(15.88)	18.44	(14.36)

Table 3.4: Coverage (Cov.) and mean (with std. dev.) absolute error (AE) in percentage points on plankton datasets with real-world shift.

		BPL	**FPL**	**OPL** ($q_c^T \leq 1\%$)	**OPL** ($q_c^T > 1\%$)
CC	Cov.	0.26%	3.77%	54.62%	1.75%
	AE	4.73 (4.50)	2.57 (2.02)	0.07 (0.04)	2.76 (2.68)
PCC	Cov.	11.26%	19.76%	70.97%	13.61%
	AE	4.55 (4.63)	2.53 (2.11)	0.08 (0.04)	2.68 (2.82)
EM	Cov.	17.80%	29.50%	46.64%	25.29%
	AE	6.15 (6.47)	3.52 (4.08)	0.11 (0.08)	3.73 (5.14)
EM-BS	Cov.	22.12%	29.84%	79.25%	29.50%
GSLS	Cov.	87.63%	51.44%	0.33%	53.91%
	AE	7.79 (8.61)	10.82 (7.18)	14.07 (6.49)	10.40 (7.25)

that EM and GSLS may incorrectly adjust intervals away from extremely low or zero values and towards higher values, which may be inappropriate in applications with high class imbalance. The bootstrapped prediction intervals of EM-BS address this low coverage for low prevalence targets but otherwise have similar or only slightly better performance than the native intervals. Given that neither PCC nor EM achieves a reasonable degree of coverage and that CC and PCC have the lowest overall error, there appears to be non-prior general dataset shift between the training and test samples in all three datasets. This is particularly interesting given that some prior-shift assuming quantification methods improved quantification performance in the previous study with this dataset (González et al., 2019), suggesting such methods may still provide benefits in some cases of general shift. However, those quantification methods cannot produce prediction intervals; therefore, they are not studied further here.

The results in Tables 3.2, 3.3 and 3.4 demonstrate that while the proposed GSLS quantification model does not improve quantification error over other methods, it does provide prediction intervals that are more reliable under a wider range of shift conditions than either PCC or EM. However, Table 3.2 demonstrates cases where GSLS

may provide inadequate prediction intervals due to underestimation of shift parameters w^+ and w^-, which are analysed further in the following subsection. Additionally, the reasonably similar coverage of native EM intervals and bootstrapped intervals (EM-BS) demonstrates that it is reasonable to continue experiments with the native intervals alone, which are supported in the proposed framework for constrained quantification intervals.

3.6.3 Analysis of Shift Misestimation

To better understand situations where shift is misestimated when fitting GSLS, Fig. 3.4 compares estimated shift against the degree of shift determined by the parameters of experiments with introduced shift. The plotted w^R represents the weight of the remaining distribution shared by the source and target, reflecting the combined effect of loss and gain weights in a single value: $w^R = (1 - w^+)(1 - w^-)$. Therefore, a low w^R indicates a high degree of shift.

Fig. 3.4 shows that in cases of little or no shift (w^R close to 1), the degree of shift tends to be overestimated ($\hat{w}^R < w^R$). In such cases, sample noise causes differences between source and target histograms that are misinterpreted as shift. In order to avoid applying GSLS in such cases without shift, a statistical test can be used first to check for the presence of any dataset shift. Fig. 3.4 also shows that large shifts (w^R close to 0) tend to be underestimated ($\hat{w}^R > w^R$). This explains the poor coverage of GSLS intervals for very large shifts ($w^+ = 1$ or $w^- = 1$) in Table 3.2. Such cases reflect the problem of "explaining" large shifts as combinations of smaller shifts. As discussed in Section 3.3.2, such extreme shifts would not typically be expected in a practical predictive context.

The degree of underestimation also varies between datasets in Fig 3.4. Underestimation is greater for HLA than similar dataset HLL for large shifts, reflected in their

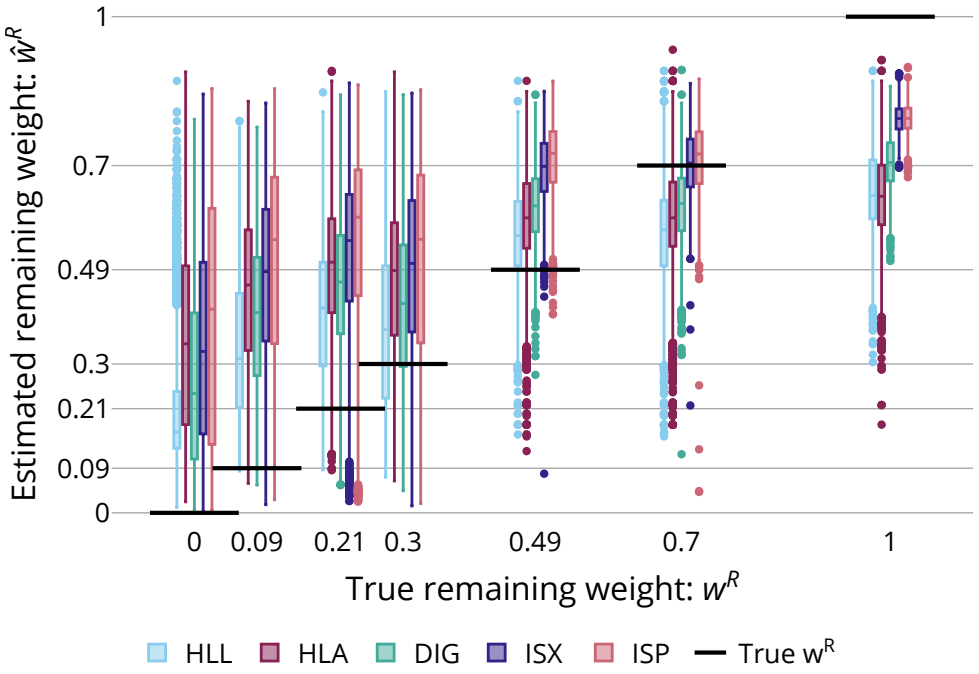


Figure 3.4: The remaining weight tends to be overestimated when there is little shift, and underestimated under extreme shift.

differing GSLS coverage (Table 3.2). The greatest underestimation is seen for ISP, which suffered from the worst GSLS coverage. This indicates some unobservable concept shift ($P^S(X) = P^T(X)$, $P^S(Y|X) \neq P^T(Y|X)$) between the concepts of both HLA and ISP, as discussed in Section 3.3.2. However, ISX has similar dataset properties to ISP and similarly high underestimation, yet GSLS produces adequate intervals for ISX.

To explain this difference, Fig. 3.5 plots the difference between the true and estimated remaining weights ($w^R - \hat{w}^R$) against the *discrepancy metric*. Inspired by the discrepancy ratio (Møller, Weedon-Fekjær & Haldorsen, 2005), the discrepancy metric compares the difference between the true class proportion q_c^T and the interval's midpoint to the interval's width:

$$\text{discrepancy} = \frac{|\text{midpoint} - q_c^T|}{\text{width}/2} \tag{3.35}$$

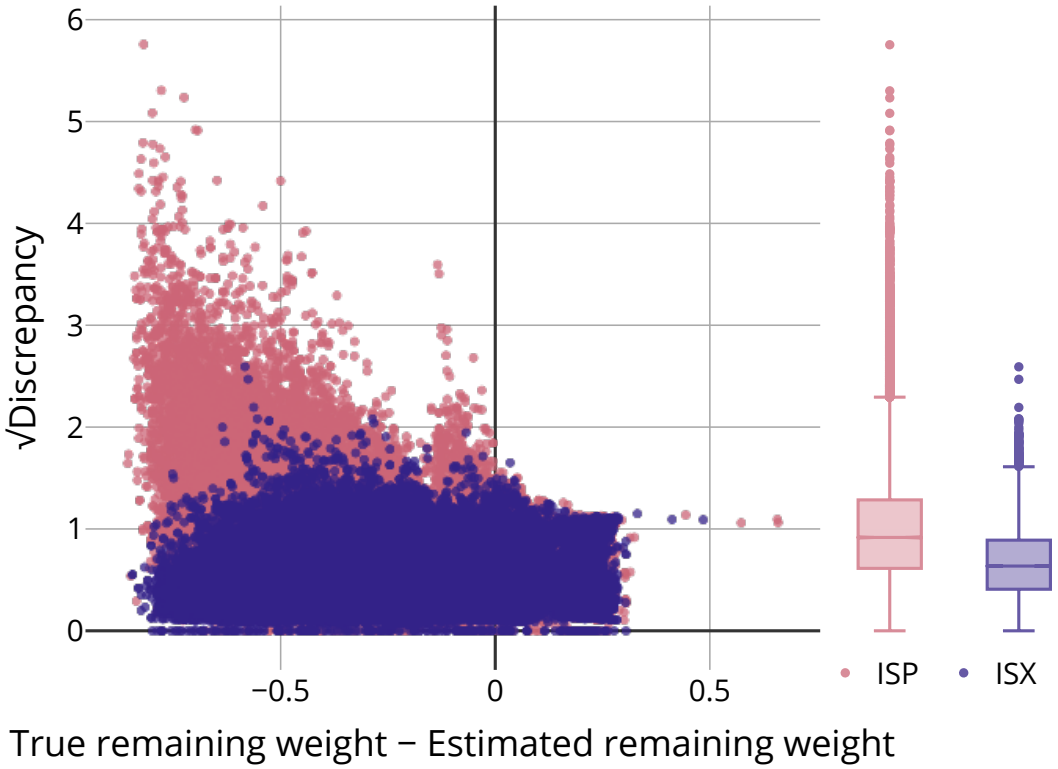


Figure 3.5: Shift is underestimated for both ISX and ISP, but only the prediction intervals of ISP are negatively impacted.

Note that $discrepancy$ <= 1 when q_c^T falls within the interval, and is greater otherwise. It can be seen from Fig. 3.5 (where discrepancy is scaled as a square root) that although ISX and ISP similarly underestimate the remaining weight, only the prediction intervals of ISP are negatively impacted. This indicates that the similarity between feature distributions $P(X)$ of ISX concepts actually reflects similar classification rules $P(Y|X)$, and GSLS is accurately modelling an *effective* shift less than that indicated by parameter-based w^R.

Table 3.5 further demonstrates that, in such cases where the effective shift is less than that indicated by parameters (e.g. DIG and ISX), GSLS intervals can be significantly shorter than TGSLS intervals while still achieving adequate coverage (Table 3.2). While some GSLS intervals in Table 3.5 may seem too wide to provide useful information about the true class proportion, they may still be useful to inform the user of the degree

Table 3.5: Mean (with std. dev.) 80% prediction interval widths in percentage points. PCC and EM are omitted, as their intervals frequently have inadequate coverage. Bold indicates when one of GSLS or TGSLS is significantly narrower than the other (by a corrected re-sampled t-test; $\alpha = 0.05$).

	$\{w^+, w^-\}$	$\{0,0\}$	$\{0,.3\}$	$\{0,.7\}$	$\{0,1\}$	$\{.3,0\}$	$\{.3,.3\}$	$\{.3,.7\}$	$\{.3,1\}$	$\{.7,0\}$	$\{.7,.3\}$	$\{.7,.7\}$	$\{.7,1\}$	$\{1,0\}$	$\{1,.3\}$	$\{1,.7\}$	$\{1,1\}$
HLL	GSLS	25 (7)	28 (7)	**39 (11)**	67 (14)	34 (7)	32 (7)	43 (12)	**68 (12)**	54 (11)	47 (12)	52 (14)	**68 (12)**	66 (14)	**60 (15)**	**59 (15)**	**66 (14)**
	TGSLS	**2 (1)**	26 (6)	59 (15)	82 (<1)	27 (<1)	33 (2)	52 (5)	82 (<1)	56 (<1)	57 (<1)	59 (<1)	82 (<1)	82 (<1)	82 (<1)	82 (<1)	82 (<1)
HLA	GSLS	18 (2)	19 (2)	**24 (4)**	**35 (7)**	**22 (3)**	**21 (3)**	**24 (4)**	**36 (6)**	**34 (6)**	**30 (6)**	**31 (6)**	**36 (6)**	**35 (7)**	**32 (6)**	**32 (6)**	**35 (7)**
	TGSLS	**5 (<1)**	**13 (<1)**	30 (2)	82 (<1)	27 (<1)	28 (<1)	35 (<1)	82 (<1)	56 (<1)	57 (<1)	57 (<1)	82 (<1)	82 (<1)	82 (<1)	82 (<1)	82 (<1)
DIG	GSLS	19 (6)	23 (8)	36 (17)	**54 (20)**	26 (7)	26 (7)	38 (15)	**54 (17)**	44 (15)	**39 (13)**	44 (16)	**55 (17)**	**54 (19)**	**50 (19)**	**51 (19)**	**55 (19)**
	TGSLS	**3 (<1)**	25 (11)	58 (25)	80 (<1)	25 (<1)	32 (3)	53 (10)	80 (<1)	57 (<1)	57 (<1)	59 (1)	80 (<1)	80 (<1)	80 (<1)	80 (<1)	80 (<1)
ISX	GSLS	12 (3)	16 (5)	36 (20)	**51 (20)**	19 (5)	**19 (5)**	33 (16)	**48 (17)**	**36 (13)**	**32 (12)**	**37 (16)**	**48 (17)**	**50 (20)**	**47 (20)**	**47 (20)**	**51 (20)**
	TGSLS	**1 (<1)**	25 (11)	60 (25)	80 (<1)	24 (<1)	31 (3)	53 (9)	80 (<1)	56 (<1)	56 (<1)	59 (1)	80 (<1)	80 (<1)	80 (<1)	80 (<1)	80 (<1)
ISP	GSLS	12 (2)	17 (5)	37 (21)	**47 (23)**	**18 (5)**	**18 (5)**	33 (18)	**42 (22)**	**33 (15)**	**29 (13)**	**35 (19)**	**42 (21)**	**46 (23)**	**43 (22)**	**43 (22)**	**45 (23)**
	TGSLS	**3 (<1)**	27 (10)	63 (22)	80 (<1)	24 (<1)	31 (3)	53 (10)	80 (<1)	56 (<1)	56 (<1)	58 (1)	80 (<1)	80 (<1)	80 (<1)	80 (<1)	80 (<1)

of quantification uncertainty. Furthermore, they can still be constrained to user-specified limits through the framework presented in Section 3.5.

Finally, Fig. 3.4 shows a tendency of slightly greater $\hat{w}^R$ for combinations of gain and loss shift ($w^R \in \{0.09, 0.21, 0.49\}$), reflecting the potential for shift underestimation when $P^+(X)$ and $P^-(X)$ overlap (discussed in Section 3.3.2). Overlap between class-conditional feature distributions may also cause shift underestimation for ISP under prior shift and potentially contribute to underestimation under GSLS shift.

Overall, these observations highlight the effectiveness of GSLS in producing reliable prediction intervals under realistic shift conditions.

3.6.4 Histogram Bins Sensitivity Analysis

Further experiments were performed to analyse the sensitivity of the GSLS model to the number of histogram bins b (its primary hyperparameter), presented in Fig. 3.6 and Table 3.6. Fixed bin counts of 5 and 50 were compared to the *auto* value set by the rule-of-thumb used up to this point: $2\min(n^S, n^T)^{2/5}$. Given the interaction between the choice of b and sample noise, as well as the importance of evaluating quantification on different target sample sizes (A. Maletzke et al., 2020), experiments were performed on variants of ISX with varied target sample sizes ($n^T = 500$ for ISX-500, etc.) for 100 random source and target sample pairs (fewer random seedings than prior experiments, as these experiments were also run serially for a runtime evaluation).

Fig. 3.6 plots the mean $\hat{w}^R$ for varied bin counts and target sample sizes. As expected, $\hat{w}^R$ decreases with more bins, as the increased granularity results in source and target histograms that are more dissimilar. $\hat{w}^R$ also decreases with fewer target instances, particularly under low shift, where sample noise has a greater effect on GSLS fitting. Table 3.6 shows a moderate increase in runtime with increased bins and, to a lesser extent, more target instances. Overall, the rule-of-thumb achieves a reasonable

Table 3.6: Mean (with std. dev.) of GSLS runtimes (in msec).

	5 bins		**Auto bins**		**50 bins**	
ISX-50	93.9	(14.8)	141.1	(23.8)	310.6	(74.7)
ISX-250	139.6	(48.6)	262.7	(55.2)	587.9	(155.3)
ISX-500	217.9	(122.3)	362.5	(72.7)	666.6	(155.6)

trade-off between overestimating small shifts and underestimating extreme shifts. Other choices for b achieve similar results, except for severe shift overestimation with 50 bins for 50 target instances, which poorly models differences between source and target distributions.

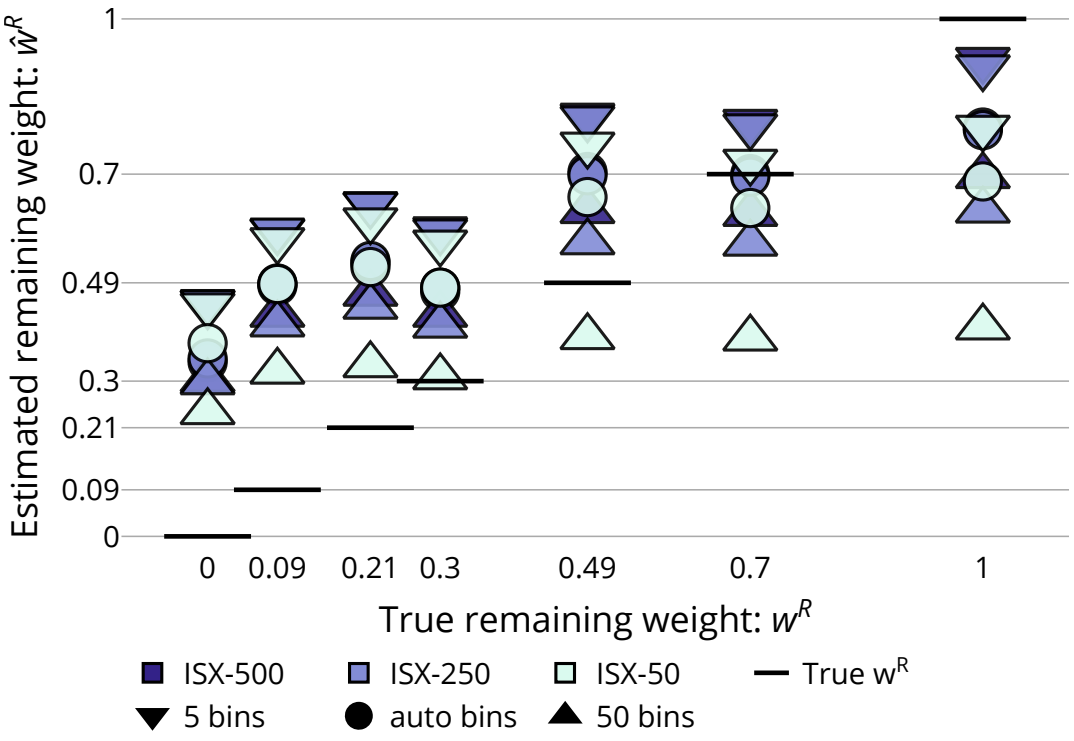


Figure 3.6: Mean $\hat{w}^R$ for different bin and target instance counts.

3.6.5 Shift Detection Experiments

To identify the most suitable shift tests for dynamically selecting a quantification method, Table 3.7 evaluates how often each of the *any shift* and *non-prior shift* tests presented in Section 3.4 detects shift under the same conditions of shift as in Tables 3.2 and 3.3. A threshold of $\alpha = 0.05$ was set for rejecting the null hypothesis, which is approximately equivalent to the 2-standard-deviation threshold used by the original CDT test under its assumption of a normal distribution (A. Maletzke et al., 2018). Of the two tests for any shift, KS more accurately detects GSLS shift, while LR more accurately detects prior shift. However, KS is less likely to detect shift when no shift has occurred, making it a more reliable test overall. Of the tests for non-prior shift, WPA and CDT variants with KS and HD distances are able to detect non-prior GSLS shift successfully, but they also incorrectly fire under conditions of prior shift and even no shift reasonably often. The AKS test proposed in this research is less sensitive to GSLS shift than WPA and CDT, but it is also less likely to fire under conditions of prior and no shift.

To evaluate the overall performance of dynamically selecting a quantification method, Fig. 3.7 plots the coverage and error achieved by static PCC, EM, and GSLS quantifiers against dynamic selections based on different combinations of *any shift* and *non-prior shift* tests using the decision tree presented in Section 3.4. Absolute error is plotted against inverse coverage such that an optimal quantifier will be located in the bottom left corner. Metric values are squared to prefer strategies that achieve balanced performance across experiments, and values are pre-aggregated by the 15 combinations of dataset and shift in Table 3.7 to ensure that each scenario has an equal weight in the final mean. Finally, as the experiments in Section 3.6.2 showed that PCC achieves lower error under GSLS shift than GSLS estimates, the dynamic quantification methods use PCC point estimates with GSLS prediction intervals when general shift is detected. Fig. 3.7 shows that most of the dynamic quantifiers achieve lower error than any static

Table 3.7: Frequency of shift tests detecting shift under varied shift conditions.

	Shift	"Any" tests KS	"Any" tests LR	"Non-prior" tests WPA-KS	WPA-HD	CDT-KS	CDT-HD	AKS
HLL	None	2%	18%	23%	27%	17%	29%	0%
	GSLS	81%	75%	90%	86%	88%	85%	61%
	Prior	71%	89%	59%	44%	56%	39%	16%
HLA	None	4%	46%	40%	53%	17%	28%	6%
	GSLS	87%	94%	88%	85%	82%	83%	84%
	Prior	65%	96%	69%	76%	50%	62%	41%
DIG	None	4%	13%	21%	32%	14%	21%	0%
	GSLS	83%	71%	84%	83%	81%	79%	57%
	Prior	77%	82%	44%	33%	43%	27%	16%
ISX	None	4%	18%	19%	56%	12%	38%	0%
	GSLS	93%	84%	89%	94%	86%	90%	68%
	Prior	88%	86%	42%	53%	39%	44%	14%
ISP	None	4%	13%	23%	53%	10%	34%	0%
	GSLS	91%	87%	84%	91%	78%	85%	61%
	Prior	84%	89%	49%	72%	38%	58%	16%

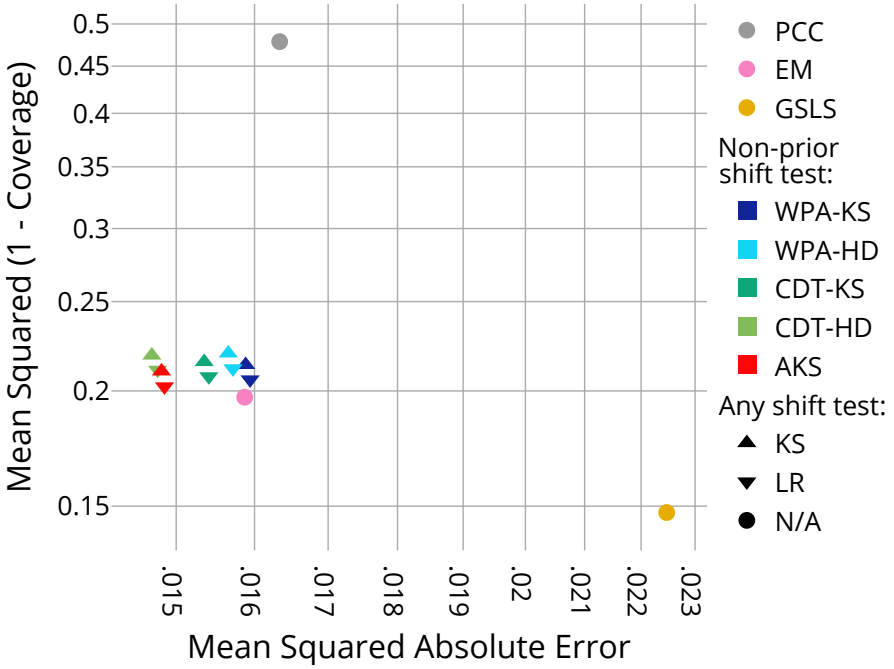


Figure 3.7: Comparison of coverage and absolute error for quantification method selection based on different combinations of shift tests.

quantifier while achieving almost as much coverage as EM and GSLS. However, rank differences in both metrics among the dynamic quantifiers and the static EM quantifier were not statistically significant according to a Friedman test and accompanying Nemenyi post hoc test ($\alpha = 0.05$).

The results in Table 3.7 and Fig. 3.7 have shown that KS is generally more reliable than LR for detecting any shift and that similar coverage and error can be achieved with any of the non-prior shift tests. To further differentiate the tests, a runtime evaluation was performed over 100 experiments with no shift on the ISX dataset, excluding the time to perform EM quantification between observed source and target samples. AKS was found to be substantially faster ($\mu = 1.9$msec$, \sigma = 0.4$msec) than the next fastest test for non-prior shift (WPA-HD: $\mu = 481.3$msec$, \sigma = 36.2$msec), as it does not need to expensively compute distribution distances for simulated samples (though such computations only need to be repeated for CDT when the source sample changes). Both tests for any shift were faster than all tests for non-prior shift ($\mu \leq 0.7$msec$, \sigma \leq 0.2$msec). Such performance differences will be important in applications with larger source and target sample sizes and when a large number of target samples must be evaluated.

The results presented above suggest that a combination of a KS test for any shift followed by an AKS test for non-prior shift provides the best balance of quantification and runtime performance. Table 3.8 presents the performance of this dynamic quantification method under each shift condition, showing that it achieves reasonable coverage on most datasets and error close to the best method in Table 3.3 under each shift condition. Additionally, both KS and AKS detected shift in $\geq 96\%$ of test samples for each of the three plankton datasets, supporting the earlier conclusion that the dataset contains non-prior, general shift.

Table 3.8: Coverage (Cov.) and mean (with std. dev.) absolute error (AE) in percentage points for the KS+AKS dynamic quantification method.

		No Shift	**GSLS Shift**	**Prior Shift**
HLL	Cov.	83%	87%	91%
	AE	1.14 (1.02)	7.92 (9.28)	2.68 (5.06)
HLA	Cov.	82%	72%	80%
	AE	2.29 (0.70)	8.32 (3.71)	4.36 (1.77)
DIG	Cov.	96%	75%	92%
	AE	0.75 (0.65)	9.48 (11.72)	3.96 (8.33)
ISX	Cov.	81%	78%	91%
	AE	0.47 (0.42)	8.58 (11.97)	2.24 (5.75)
ISP	Cov.	74%	42%	62%
	AE	1.30 (1.14)	21.77 (18.97)	9.10 (16.85)

3.6.6 Constrained Quantification Interval Experiments

As no prior work has constrained quantification intervals through rejection, the proposed framework based on binary integer programming (referred to as BIP) was compared to two baseline approaches. The first baseline (PT) performs "confidence-based rejection" (Ni et al., 2019) by performing a binary search for a confidence threshold t such that rejecting the set of instances $\left\{x^T_{(j)} \in X^T_{(j)}, \left(\max_c g_c(x^T_{(j)})\right) < t\right\}$ results in interval widths within user-specified limits. At each search step, interval widths are computed by applying the quantification method to the selected subset without re-estimating invariant target distribution parameters (such as EM-adjusted probabilities or the GSLS loss weight). The second baseline (APT) is similar to PT, but it computes interval widths using the same approximate quantification distribution $\hat{P}(q^T_{(j)})$ and variance $\hat{V}(q^T_{(j,\delta)})$ as BIP. This second baseline will reveal which differences between PT and BIP are due to these approximations (i.e. when PT and APT perform differently) or due to differences between BIP-optimisation and confidence-based rejection (i.e. when APT and BIP perform differently).

Each of these methods was tested on 1,000 randomly seeded source and target samples from each of the benchmark datasets, constraining PCC intervals under no shift, EM intervals under prior shift, and GSLS intervals under a moderate degree of GSLS shift ($w^- = 0.3, w^+ = 0.3$). Experiments were also performed for all three quantification methods with the plankton datasets, but only for test samples containing 4,000 or fewer instances due to limitations of the community licence for the FICO Xpress BIP solver. In order to set a relatively consistent task across settings, each experiment aims to constrain each class's interval over all target instances ($m = k$ and $\{X^j, Y^j\} = \{X^T, Y^T\}\ \forall\ j$) to be no wider than half of the original maximum width for any class: $\bar{\omega} = \frac{1}{2}\max\left(u_j - l_j\ \forall\ j\right)$ where $P(l_j \leq q^T_{(j)} \leq u_j) = 0.8$. Post-rejection intervals are produced by the same method used to compute intervals at each step of PT.

Table 3.9 presents the results of the interval constraining experiments, with bold font used to highlight mean statistics significantly different to the BIP result for the same experimental setting according to a t-test[6] at $\alpha = 0.05$. The *Rejection* column reports the mean proportion of target instances that were rejected to satisfy the interval limits. It shows that BIP requires less rejection in most situations, especially for multi-class datasets where BIP minimises rejection by optimising interval limits for all classes simultaneously. An exception to this trend is seen for PCC intervals on the benchmark datasets and BPL, where PT requires less rejection than the approximate methods. This can be mainly attributed to the fact that a probability threshold is the optimal rejection strategy for PCC intervals in binary datasets. Note that constraining intervals to half of the original maximum width sometimes requires rejecting more than half of the target sample, particularly for EM quantification. This is to be expected, as the interval width does not depend linearly on the number of target instances. For example, a simple binomial distribution's standard deviation (and therefore interval width) is proportional

[6]Or a corrected re-sampled t-test (Nadeau & Bengio, 1999) for the benchmark dataset experiments where source and target samples are re-sampled.

Table 3.9: Comparison of methods for constraining quantification intervals on benchmark datasets with varied shift conditions and plankton datasets with real-world shift. Statistics are given in percentage points and reported as means with (std. dev.) where applicable.

		PCC Intervals				EM Intervals				GSLS Intervals			
		Rejection	**Limit Δ**	**Cov. Δ**	**Int. Error**	**Rejection**	**Limit Δ**	**Cov. Δ**	**Int. Error**	**Rejection**	**Limit Δ**	**Cov. Δ**	**Int. Error**
HLL	BIP	33.5 (27.0)	1.2 (0.8)	-45.6	3.2 (3.1)	58.0 (7.9)	-1.4 (1.4)	-30.2	3.7 (5.9)	43.7 (6.5)	0.2 (4.1)	-19.2	1.4 (2.4)
	APT	34.2 (27.2)	1.3 (0.8)	-46.6	3.6 (3.7)	**79.8 (14.1)**	1.5 (2.6)	-21.2	1.4 (2.7)	47.7 (7.1)	1.5 (5.5)	-25.8	1.9 (2.8)
	PT	4.8 (3.9)	0.6 (0.6)	-31.3	1.5 (1.5)	**74.1 (9.1)**	0.4 (0.6)	-28.2	2.3 (3.7)	45.6 (11.1)	0.8 (0.9)	-23.2	1.6 (2.4)
HLA	BIP	31.7 (6.3)	-0.2 (0.7)	-28.2	1.7 (0.8)	58.7 (9.1)	-1.7 (1.7)	1.8	0.4 (0.6)	39.4 (6.6)	-1.3 (2.6)	-3.6	0.4 (0.4)
	APT	**37.2 (7.5)**	0.3 (0.9)	-40.5	2.4 (1.0)	**91.0 (2.9)**	**4.8 (0.9)**	4.9	0.2 (0.2)	**48.3 (6.7)**	-2.4 (2.8)	-7.8	0.6 (0.5)
	PT	30.1 (9.6)	0.4 (0.5)	-34.9	2.1 (1.0)	**80.4 (4.9)**	**0.6 (0.6)**	-1.0	0.4 (0.4)	**54.9 (9.7)**	0.7 (0.8)	-10.1	0.6 (0.5)
DIG	BIP	21.0 (7.0)	0.3 (0.4)	-74.5	2.7 (2.5)	52.7 (6.9)	-0.5 (0.8)	-65.8	6.7 (7.9)	40.4 (8.9)	0.9 (2.4)	-36.0	2.3 (4.1)
	APT	21.0 (7.0)	0.3 (0.4)	-74.5	2.7 (2.5)	68.1 (15.7)	-0.1 (1.1)	-28.9	1.7 (3.6)	47.7 (10.2)	0.9 (2.5)	-28.1	1.6 (3.3)
	PT	**13.1 (7.8)**	0.2 (0.2)	-71.8	2.2 (2.2)	**72.5 (12.0)**	**0.4 (0.4)**	-32.5	1.9 (4.0)	44.5 (13.8)	0.4 (0.4)	-25.9	1.4 (3.1)
ISX	BIP	15.9 (7.1)	0.0 (0.1)	-72.9	2.8 (2.2)	54.3 (6.5)	-0.0 (0.4)	-89.5	10.1 (8.0)	36.6 (12.3)	0.4 (2.3)	-51.6	3.2 (4.3)
	APT	15.9 (7.1)	0.0 (0.1)	-72.9	2.8 (2.2)	69.8 (12.6)	-0.1 (0.3)	-54.4	2.1 (3.4)	43.8 (12.7)	0.4 (1.8)	-36.6	1.7 (3.0)
	PT	11.8 (6.5)	0.1 (0.0)	-71.5	2.4 (2.0)	**71.9 (9.8)**	0.1 (0.1)	-58.2	2.1 (3.4)	41.1 (12.0)	0.1 (0.1)	-36.8	1.8 (3.0)
ISP	BIP	35.9 (11.3)	0.0 (0.1)	-57.0	3.0 (2.8)	52.4 (3.2)	-0.2 (0.1)	-53.2	3.9 (4.4)	41.6 (9.2)	0.8 (1.3)	-25.8	4.1 (4.8)
	APT	35.9 (11.3)	0.0 (0.1)	-57.0	3.0 (2.8)	61.4 (12.3)	0.0 (0.5)	-43.1	3.8 (4.8)	54.3 (10.5)	0.6 (1.2)	-14.8	2.9 (4.2)
	PT	34.3 (11.8)	0.1 (0.0)	-58.3	3.0 (2.7)	60.9 (8.7)	**0.1 (0.1)**	-47.1	4.1 (4.9)	50.5 (13.3)	0.1 (0.1)	-13.9	3.0 (4.4)
BPL	BIP	25.6 (6.6)	0.0 (0.1)	12.7	0.9 (1.5)	54.9 (4.6)	-0.1 (0.2)	9.9	1.3 (2.0)	34.8 (13.5)	0.1 (2.3)	0.5	0.3 (1.3)
	APT	25.6 (6.6)	0.0 (0.1)	12.7	0.9 (1.5)	**65.8 (7.6)**	**-0.0 (0.3)**	61.3	**0.1 (0.5)**	**54.8 (15.9)**	0.2 (1.3)	-1.9	0.3 (0.7)
	PT	**24.6 (6.9)**	**0.0 (0.1)**	11.7	**0.9 (1.5)**	**66.2 (5.2)**	**0.0 (0.1)**	63.0	**0.1 (0.4)**	**53.3 (16.1)**	0.0 (0.1)	-1.6	0.3 (0.7)
FPL	BIP	26.5 (6.9)	0.0 (0.1)	13.8	0.6 (0.7)	57.6 (7.8)	0.0 (0.4)	8.6	0.9 (1.4)	35.9 (10.4)	0.0 (2.3)	1.7	0.6 (0.6)
	APT	**27.7 (7.1)**	0.0 (0.1)	17.6	**0.5 (0.7)**	**92.4 (3.1)**	**0.7 (0.5)**	25.8	**0.0 (0.1)**	**53.4 (13.1)**	**0.3 (1.2)**	-14.1	**0.8 (0.7)**
	PT	**27.0 (7.3)**	**0.0 (0.1)**	16.2	0.6 (0.7)	**80.1 (7.4)**	0.0 (0.0)	17.6	**0.2 (0.4)**	**51.9 (13.5)**	0.0 (0.1)	-13.8	**0.8 (0.6)**
OPL	BIP	26.0 (7.2)	-0.0 (0.1)	5.4	0.1 (0.1)	88.8 (4.6)	0.7 (0.5)	13.7	0.1 (0.1)	40.5 (9.3)	0.3 (2.8)	2.6	1.2 (0.6)
	APT	**31.2 (8.1)**	**0.0 (0.1)**	13.8	**0.1 (0.1)**	**91.0 (1.1)**	**0.8 (0.4)**	-21.7	**0.0 (0.0)**	**51.5 (5.7)**	0.0 (0.5)	-2.2	**1.3 (0.7)**
	PT	**30.4 (8.5)**	**0.0 (0.1)**	13.5	**0.1 (0.1)**	**76.8 (2.9)**	**0.0 (0.1)**	-29.5	**0.1 (0.1)**	**51.5 (5.7)**	0.0 (0.1)	-2.1	**1.3 (0.7)**

to the square root of its sample size. BIP rejection can still be considered a practical approach for constraining intervals, given that it requires less rejection than applying the common practice of confidence-based rejection (Ni et al., 2019).

Table 3.9 also reports on the performance of the constrained intervals themselves. *Limit* Δ reports the mean maximum difference between the constrained interval width for a class and its width limit, with negative values indicating that the constrained interval was wider than the limit. While its approximations mean that its intervals are not guaranteed to fall perfectly within the limit, the BIP intervals for these experiments are close enough for practical purposes, with only three experimental settings exceeding the limit by a mean of more than one percentage point. *Cov.* Δ reports the percentage-point coverage difference between the original and constrained intervals; a negative Δ indicates that the constrained intervals cover fewer of the true class proportion than the original intervals. While BIP intervals improve overall coverage on the plankton datasets, there is often a large decrease in coverage on the benchmark datasets. This decrease in coverage also occurs with PT and APT and is most extreme for the PCC and EM intervals that are heavily dependent on instance classification probabilities. This suggests that the high probabilities of selected instances are calibrated more over-optimistically than those being rejected, resulting in intervals that are too narrow. This highlights the importance of well-calibrated probabilities for producing accurate intervals, especially for high-confidence classifications if the low-confidence classifications are to be rejected. Nonetheless, the true proportions are still generally near the edges of the intervals, as shown by the reasonably low *Int. Error* values, which measure the mean percentage-point distance from the true proportion to the interval edge (or zero if the true proportion is within the interval).

Not only does BIP typically requires less rejection to achieve similar intervals to the baseline approaches, but it can also run the optimisation within a practical amount of time. Table 3.10 reports runtime results of constraining intervals for 100 test samples of

Table 3.10: Comparison of method runtimes for constraining quantification intervals reported as median seconds with (std. dev.).

		PCC	EM	GSLS
BPL	BIP	0.04 (0.01)	2.50 (2.01)	1.97 (1.40)
	APT	**0.01 (0.00)**	**0.01 (0.00)**	**0.39 (0.21)**
	PT	**1.53 (1.00)**	**0.01 (0.00)**	**2.56 (4.84)**
FPL	BIP	0.11 (0.06)	1.77 (0.90)	6.27 (8.23)
	APT	**0.45 (0.22)**	**0.46 (0.23)**	**1.43 (0.26)**
	PT	**3.09 (2.38)**	**0.67 (0.32)**	**11.19 (14.82)**
OPL	BIP	0.53 (0.31)	5.95 (60.61)	104.93 (122.79)
	APT	**4.24 (0.80)**	**4.98 (0.73)**	**10.57 (1.58)**
	PT	**25.59 (22.63)**	**7.10 (1.03)**	137.39 (94.12)

the plankton datasets, where the larger test sample sizes ($\mu = 2,337, \sigma = 885$) highlight how the methods perform at scales where constant overheads (particularly for BIP) are less relevant. Runtimes significantly different to BIP according to a t-test ($\alpha = 0.05$) are highlighted in bold. Median runtimes are reported to lessen the impact of outliers, as BIP optimisation can run exceptionally slowly in rare cases. Observe that BIP is faster than PT, except for constraining EM intervals on the BPL and FPL datasets. Because these datasets have fewer classes, the interval widths can be computed rapidly in PT's binary search. BIP is even faster than the approximate binary search method APT for constraining PCC intervals on the multi-class FPL and BPL datasets, as APT re-computes intervals for all classes at each search step.

3.6.7 Synthetic Example Experiments

In order to evaluate the methods for producing and constraining quantification intervals without the confounding factor of imperfectly calibrated classification probabilities, this section performs experiments on synthetic datasets with known probability distributions.

Three synthetic datasets simulate conditions of no shift (SNS), prior shift (SPS), and GSLS shift (SGS), respectively. Each dataset has two possible output classes

($Y \in \{0, 1\}$) and only a single input feature ($X \in \mathbb{R}^1$) drawn from a mixture of Gaussian distributions. A single 1000-record training sample $\{X^S, Y^S\}$ is drawn from a source distribution $P^S(X, Y)$ and one thousand 100-record test samples $\{X^T, Y^T\}$ are drawn from a target distribution $P^T(X, Y)$. Sampling is performed according to the following probability distributions for each dataset:

1. Synthetic "No Shift" Dataset (SNS):
    - The class for each instance is drawn from the same distribution for both the source and target samples:
        - $P^S(Y = 1) = P^T(Y = 1) = 0.5$
    - The input feature for each instance is drawn from a distribution that is conditional on its class, with the same class-conditional distribution used for both the target and source samples:
        - $P^S(X|Y = 0) = P^T(X|Y = 0) = \mathcal{N}(0.2, 0.3)$
        - $P^S(X|Y = 1) = P^T(X|Y = 1) = \mathcal{N}(0.8, 0.3)$
2. Synthetic "Prior Shift" Dataset (SPS):
    - The class for each instance is drawn from a different distribution depending on whether it belongs to the source or target sample:
        - $P^S(Y = 1) = 0.5$
        - $P^T(Y = 1) = 0.75$
    - The input feature for each instance is drawn from a distribution that is conditional on its class, with the same class-conditional distribution used for both the target and source samples:
        - $P^S(X|Y = 0) = P^T(X|Y = 0) = \mathcal{N}(0.2, 0.3)$
        - $P^S(X|Y = 1) = P^T(X|Y = 1) = \mathcal{N}(0.8, 0.3)$
3. Synthetic "GSLS Shift" Dataset (SGS):

- Each instance's class and feature will be sampled from distributions determined by the component C that the instance belongs to, which is either the remaining (R), loss ($-$), or gain ($+$) component. The component for each instance is drawn using different priors for the source and target samples. The source distribution is a mixture of the remaining and loss components and the target distribution is a mixture of the remaining and gain components:
  - $P^S(C = R) = \frac{2}{3}$, $P^S(C = -) = \frac{1}{3}$, and $P^S(C = +) = 0$
  - $P^T(C = R) = \frac{2}{3}$, $P^T(C = -) = 0$, and $P^T(C = +) = \frac{1}{3}$
- The class for each instance is drawn from a component-conditional distribution. The loss component only contains class 0 while the gain component only contains class 1:
  - $P^R(Y = 1) = 0.5$, $P^-(Y = 1) = 0$, and $P^+(Y = 1) = 1$
- The input feature for each instance is drawn from a distribution that is conditional on its component and class. Note that the feature distributions for gain and loss components only need to be defined for the single class that each contains, and that they have minimal overlap with the remaining distribution and each other to allow shift to be detected:
  - $P^R(X|Y = 0) = \mathcal{N}(0.2, 0.3)$ and $P^R(X|Y = 1) = \mathcal{N}(0.8, 0.3)$
  - $P^-(X|Y = 0) = \mathcal{N}(-1, 0.3)$
  - $P^+(X|Y = 1) = \mathcal{N}(2, 0.3)$

In addition to evaluating quantification based on logistic regression outputs that estimate source-distribution probabilities $P^S(Y|X)$, the synthetic nature of these datasets permits experimentation with the true source-distribution probabilities, calculated as:

$$\begin{aligned} P^S(Y|X) &= \frac{P^S(X|Y)P^S(Y)}{P^S(X)} \\ \text{where: } P^S(X) &= \sum_{y \in Y} P^S(Y = y)P^S(X|Y = y) \end{aligned} \tag{3.36}$$

For the SGS dataset, $P^S(Y|X)$ can be computed based on the probabilities for each component $\theta \in \{R, -, +\}$:

$$P^S(Y|X) = \sum_{\theta \in \{R,-,+\}} P^S(C = \theta) P^\theta(Y|X) \tag{3.37}$$

where $P^\theta(Y|X)$ can be calculated similarly to $P^S(Y|X)$ in Equation (3.36).

The results in Table 3.11 show that the appropriate quantification method for each synthetic dataset achieves coverage above the intended 80% with both logistic regression outputs (LR) and the true source probabilities (TP).

Results for constrained intervals in Table 3.12 generally show similar trends to the results for the benchmark and plankton datasets in Table 3.9. However, the coverage of intervals does not tend to decrease for constrained intervals (i.e. a negative *Cov.* Δ) as it did for the other datasets. This is because the true probabilities and more accurately calibrated logistic regression probabilities on these simple datasets are not overly optimistic at their extreme values, as discussed in Section 3.6.6. A key exception to this is seen for the PT and APT methods on the SGS datasets, where coverage is still reduced. The instances that PT and APT first reject from the target sample are those nearest the decision boundary, which are primarily instances from the remaining component in the SGS dataset. Rejecting these instances before others means that the

Table 3.11: Coverage of true class proportions by 80% prediction intervals on synthetic example datasets.

Method & Dataset	**Source Probs.**	**Coverage**
PCC on SNS	LR	84%
	TP	83%
EM on SPS	LR	96%
	TP	96%
GSLS on SGS	LR	100%
	TP	82%

Table 3.12: Comparison of methods for constraining quantification intervals on synthetic example datasets. Statistics are given in percentage points and reported as means with (std. dev.) where applicable.

	PCC Intervals on SNS			**EM Intervals on SPS**			**GSLS Intervals on SGS**		
	Rejection	**Limit** Δ **Cov.**	Δ **Int. Error**	**Rejection**	**Limit** Δ **Cov.**	Δ **Int. Error**	**Rejection**	**Limit** Δ **Cov.**	Δ **Int. Error**
LR MIP	44.5 (3.1)	0.2 (0.4)	4.7 0.1 (0.5)	51.0 (1.9)	-0.9 (0.4)	-0.1 0.1 (0.3)	49.6 (3.7)	-1.3 (2.3)	-0.1 0.0 (0.1)
LR APT	44.5 (3.1)	0.2 (0.4)	4.7 0.1 (0.5)	**53.9 (3.4)**	**-0.9 (0.4)**	4.5 **0.0 (0.0)**	**63.2 (7.2)**	**-0.1 (1.6)**	-61.8 **1.1 (1.0)**
LR PT	**41.9 (4.2)**	**0.4 (0.2)**	2.5 **0.2 (0.5)**	**59.7 (4.2)**	**0.3 (0.3)**	4.4 **0.0 (0.0)**	**63.0 (8.9)**	**0.3 (0.4)**	-62.1 **1.1 (1.0)**
TP MIP	42.7 (3.2)	0.2 (0.4)	4.3 0.2 (0.5)	51.0 (2.0)	-0.9 (0.4)	1.5 0.0 (0.3)	50.1 (2.9)	2.5 (1.8)	16.2 0.0 (0.3)
TP APT	42.7 (3.2)	0.2 (0.4)	4.3 0.2 (0.5)	**54.6 (3.8)**	**-0.8 (0.5)**	4.3 **0.0 (0.0)**	**53.1 (3.5)**	**6.6 (4.4)**	16.8 0.0 (0.1)
TP PT	**39.6 (4.3)**	**0.3 (0.2)**	1.1 **0.2 (0.6)**	**60.5 (4.2)**	**0.3 (0.3)**	4.2 **0.0 (0.0)**	**44.4 (3.1)**	**0.3 (0.3)**	16.9 0.0 (0.2)

post-rejection intervals are computed based on instances from the gain component, which have source-distribution probabilities that are less well-calibrated because they are not from the source distribution. The BIP rejection method does not suffer from this particular issue because its strategy of weighting instances by their region's local gain weight results in the rejection of instances from the gain component before those near the decision boundary for the SGS dataset. It is also worth highlighting that the *Int. Error* is still low for PT and APT on the SGS dataset, so the intervals are not far from covering the true class proportions.

3.7 Conclusion

This chapter has addressed RO2 through several contributions. RO2.1 is addressed with a novel Gain-Some-Lose-Some (GSLS) model for reliable quantification under more general conditions of dataset shift than accounted for by existing quantification methods. Additionally, RO2.2 is addressed by a decision tree for dynamically selecting the most appropriate quantification method for given source and target samples. An experimental evaluation demonstrated that a combination of a KS test for any shift followed by a newly proposed AKS test for non-prior shift provides the best balance of quantification and runtime performance. Finally, RO2.3 is addressed by a framework for constraining quantification interval widths to user-specified limits by requesting class labels from the user for a smaller set of target instances than would be required with confidence-based rejection.

Chapter 4

Learning Regions of Classifications to Reject Under Class Noise

4.1 Introduction

This chapter is based on the work presented in "Null-Labelling: A Generic Approach for Learning in the Presence of Class Noise" (Denham et al., 2020), and addresses RO3:

RO3 Develop a model-agnostic rejection method to identify and reject regions of class noise in order to achieve a better trade-off between classification error and rate of rejection than confidence-based rejection.

For as long as machine learning researchers have strived to improve accuracy achieved by classifiers, noisy data has always presented a major obstacle. This has led to the development of many techniques for mitigating both *feature noise* (where feature values are partially dependent on a stochastic process) and *class noise* (where class labels are partially dependent on a stochastic process). While techniques such as feature selection and noise filtering are commonly used to mitigate feature noise, class noise has the potential to be much more disruptive to the learning process. If class

noise is more prevalent in certain regions of the input space under the so-called *noisy not at random* (NNAR) model (Frénay & Verleysen, 2013), then learning an accurate classifier for these regions may be impossible without prior knowledge of the noise mechanism. Such *regions of noise* may represent cases that human labellers found difficult to classify, or they may be inherent to the dataset. For example, when attempting to classify topography based on aerial photography, a mix of class values would be expected for the region of the input space where clouds had obscured photographs (Johnson & Iizuka, 2016). Such not-at-random class noise, which has been identified as a relatively unaddressed problem (Frénay & Verleysen, 2013), is the primary concern of this chapter.

When a dataset contains instances that cannot be accurately classified, one option is to allow the classifier to choose not to classify (i.e. *reject*) those instances. Building such a *rejecting-classifier* makes intuitive sense, as there are many situations where it is preferable for a decision maker to abstain from making a decision in the presence of uncertainty rather than giving their best guess, such as in a medical diagnosis scenario. Depending on the use case, rejected instances may be forwarded to a more resource-heavy classifier or a human for manual judgement. The field of *classification with rejection* has also been referred to as *learning with abstention* (Cortes, DeSalvo & Mohri, 2016), *selective classification* (Hanczar, 2019), and *cautious classification* (Ferri & Hernández-Orallo, 2004).

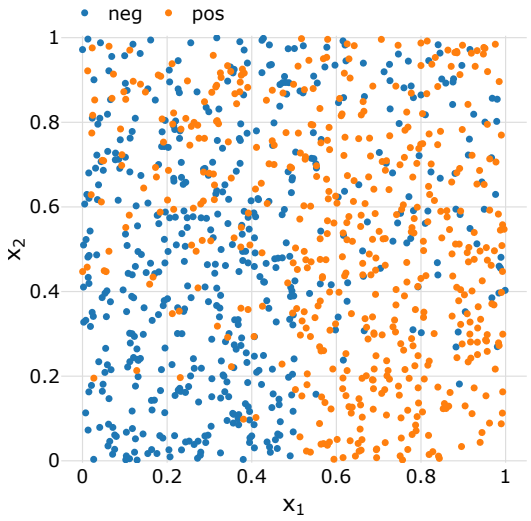
One of the most prevalent rejection methods is to reject instances where the confidence score or probability estimate provided by the classifier is below a given threshold (Fumera, Roli & Giacinto, 2000), which will be referred to as *confidence-thresholding* throughout this chapter. The popularity of this method is largely due to its simplicity of implementation for many common machine learning methods. However, confidence-thresholding may not be the optimal rejection strategy when dealing with class noise. Most classification algorithms are designed to learn optimal decision boundaries under

the assumption that every instance must be classified. Therefore, they may ignore important patterns in the dataset that indicate regions of noise if they are insufficiently powerful or otherwise do not discriminate regions of class noise. Because confidence scores are typically related to the learned decision boundaries, they may not be correlated with regions of noise in the input space.

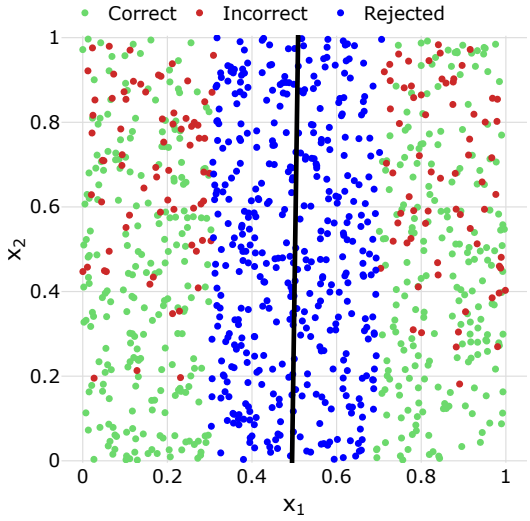
Consider the binary classification dataset in Fig. 4.1a, which contains a uniform distribution of points that originally expressed the following classification rule: $P(y|x_1 >= 0.5) = 1; P(y|x_1 < 0.5) = 0$, with $P(y)$ denoting the probability of a positive class label. However, introduced class noise replaced each class label with a random class with probability equal to x_2, such that the rule evaluates to: $P(y|x_1 >= 0.5) = 1 - \frac{x_2}{2}; P(y|x_1 < 0.5) = \frac{x_2}{2}$. Because the degree of noise is not random but is a function of x_2, this represents not-at-random class noise with a region of noise located in the upper region defined by x_2. x_1 can be said to be a *signal dimension/feature* that indicates the class value, while x_2 is a *noise dimension/feature* that indicates the degree of class noise.

Fig. 4.1b presents the classification results for a logistic regression (LR) classifier configured with confidence-thresholding to reject the 40% of instances with the lowest confidence scores. Because there is a uniform distribution of each class along the x_2 dimension, LR produces a vertical decision boundary based solely on x_1. Because LR confidence scores are determined by the distance of an instance from the decision boundary, a band of instances is rejected around this vertical decision boundary. However, this leaves a large number of errors in the region of noise (high x_2 values) that have not been rejected. It also results in the unnecessary rejection of instances with low x_2 values and x_1 values close to 0.5 that would be correctly classified without the use of rejection.

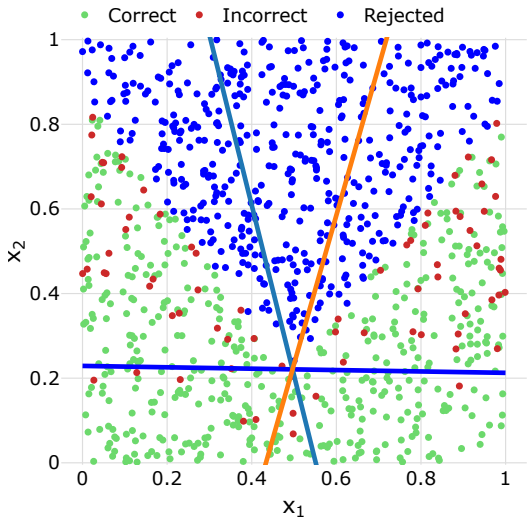
This example shows that the classifier is not taking advantage of the noise feature x_2 because it is not discriminative of class boundaries, even though it contains useful information for identifying the region of noise. Noise features like this may exist in



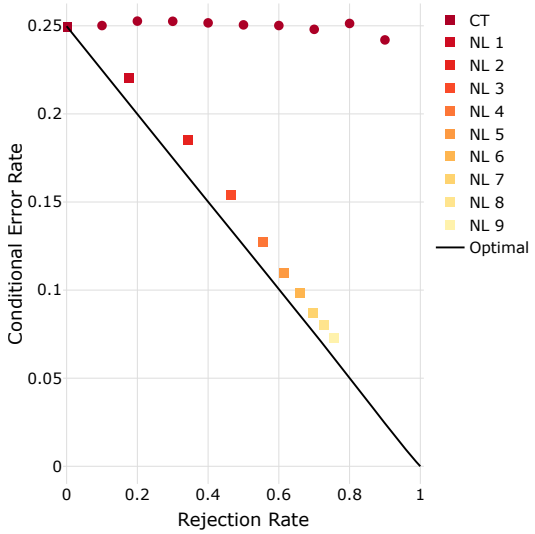
(a) Dataset with not-at-random class noise (1000 instance sample)



(b) LR Classifications with CT (with binary decision boundary)



(c) LR Classifications with NL (with class coefficient boundaries)



(d) Tradeoff plot for classifier without rejection (Base), 9 CT thresholds, and 9 NL iterations

Figure 4.1: Comparison of confidence-thresholding (CT) and null-labelling (NL) on a dataset with not-at-random class noise.

practice, such as the strength of a radio signal indicating the reliability of other signal measurements.

This chapter proposes a new rejection method called *null-labelling* that enables a classifier to learn regions of noise directly as an explicit `null` class. This is achieved by relabelling instances in the training set with a new `null` class if they are misclassified under a k-fold cross-validation scheme. Fig. 4.1c demonstrates the classification results after three iterations of null-labelling. The three lines in Fig. 4.1c represent the class-discriminating boundaries defined by the coefficients learned by multinomial logistic regression for each of the three classes: "pos", "neg", and `null`. With null-labelling, LR can learn the importance of the x_2 feature for identifying the region of noise, rejecting instances with a high x_2 value while not unnecessarily rejecting correctly classified instances near the "pos"/"neg" decision boundary.

Fig. 4.1d compares confidence-thresholding (CT) and null-labelling (NL) over 100,000 instances in terms of the tradeoff between the rejection rate and conditional error rate, which is the error rate on non-rejected (*covered*) instances (Hanczar, 2019). As the confidence threshold is raised (and the rejection rate increases), there is little reduction in error — the rejection method is not much better than randomly selecting instances to reject. However, the error is drastically reduced by applying an increasing number of null-labelling iterations, demonstrating that it is correctly rejecting instances that were being misclassified in the region of noise. Furthermore, the performance of null-labelling is near that of Bayes-optimal classification and rejection based on the true probabilities used to generate class labels. Users can use a tradeoff plot like Fig. 4.1d to identify the classifier with the optimal rejection rate for their use case, which will depend on the relative cost of a misclassification compared to handling a rejected instance in their application domain.

The null-labelling method also provides an additional benefit: by treating rejection as an explicit class, model interpretation techniques can be used to gain insights into

which regions of the input space are difficult to classify. This is seen in Fig. 4.1c, where the class boundary defined by the coefficients for the `null` class discriminates primarily on the x_2 feature, revealing the relationship between x_2 and the degree of noise. This can aid a user in diagnosing sources of class noise, which they may be able to address in order to provide a cleaner version of the training dataset, thus leading to further improvements in classification accuracy. While other rejection methods have modelled rejection as an explicit class (Cortes et al., 2016; Thulasidasan, Bhattacharya, Bilmes, Chennupati & Mohd-Yusof, 2019), they require alterations to the underlying classification algorithms and loss functions. Null-labelling, on the other hand, is a model-agnostic method that can be applied with any base classifier.

In order to produce rejecting-classifiers with rejection rates that lie between those produced by subsequent iterations of null-labelling, this chapter also presents a framework for combining rejecting-classifiers based on a unification of past theories (Ferri & Hernández-Orallo, 2004; Hanczar, 2019). This framework also supports using the capacity metric (Ferri & Hernández-Orallo, 2004) for evaluating classification performance across different rejection rates.

The following contributions are made in this chapter:

- A novel model-agnostic null-labelling method for rejection that can achieve a better tradeoff between error and rejection rates than confidence-thresholding in the presence of class noise.
- Interpretation of null-labelled models for identifying features that are correlated with class noise.
- A unification of prior theories to produce a framework for combining rejecting-classifiers and evaluating performance across different rejection rates.

The remainder of this chapter is structured as follows. Section 4.2 reviews related work on class noise and rejection while Section 4.3 presents notation for rejection and

discusses the suitability of using confidence scores for rejection. The design of the novel null-labelling method is presented in Section 4.4, and a framework for comparing rejecting-classifiers is defined in Section 4.5. Section 4.6 presents experimental results with confidence-thresholding and null-labelling, and Section 4.7 provides concluding thoughts.

4.2 Literature Review

This section provides background on class noise and its relationship to rejection. Previously proposed rejection methods are also reviewed.

4.2.1 Background on Class Noise

While the term *class noise* has been used to describe a variety of phenomena, this chapter considers class noise to be present when the class label Y for an instance depends not only on its input features X but also on a stochastic process E that distorts the relationship between X and Y. This stochastic process may describe cases where instances with the same input feature values are assigned different classes or where training instances are mislabelled (Frénay & Verleysen, 2013). Class noise can therefore be considered a generalisation of label noise, which only considers the latter case[1]. In particular, null-labelling is designed to address *noisy not-at-random* (NNAR) class noise, where E is also dependent on X — i.e. there are *regions of noise* in the input space where the stochastic process more strongly distorts the relationship between X and Y. This NNAR class noise, depicted by the statistical model in Fig. 4.2, is a generalisation of the NNAR model for label noise (Frénay & Verleysen, 2013), as it does not assume that a "true" class value underlies and determines a noisy class label.

[1]Though label noise is sometimes also referred to as class noise (Luengo, Shim, Alshomrani, Altalhi & Herrera, 2018), a distinction is made here.

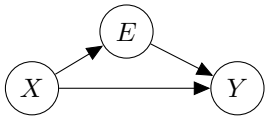


Figure 4.2: Model of NNAR class noise.

There has been extensive research into methods for mitigating label noise. Such methods rely on designing specialised classifiers that model the probability of noisy labels (Bootkrajang & Chaijaruwanich, 2018) or identifying noisy training instances so that they may be relabelled with the presumably correct class or removed from the training set entirely (Luengo et al., 2018). While such methods share iterative and label-replacing characteristics with null-labelling, they are only applicable for addressing certain cases of label noise. These methods assume the true relationship between X and Y can be learned despite label noise. However, any method to identify the instances affected by label noise must rely on prior knowledge of the noise mechanism (García-Zattera, Mutsvari, Jara, Declerck & Lesaffre, 2010), and errors due to label noise are therefore irreducible if such prior knowledge cannot be obtained or safely assumed. Furthermore, in the more general case of class noise, the stochastic process may affect the true Y values, making it impossible to learn a relationship between X and Y for some regions of the input space. It is these irreducible errors that necessitate rejection, where the classifier is allowed to reject instances expected to have a high probability of misclassification.

4.2.2 Related Work on Rejection

Rejection has recently been applied as a means of addressing label noise (Shah & Manwani, 2019; Thulasidasan et al., 2019), but it has been extensively researched prior to this. Rejection dates back at least to Chow's rule (Chow, 1970), which rejects instances with classification probabilities below a threshold determined by the relative

costs of rejection and misclassification in the application domain. Chow's rule is Bayes-optimal when the classification probabilities are perfect a posteriori probabilities (Fumera et al., 2000), but typically only probability estimates based on confidence scores are available in practice. This has led to the further refinement of rejection rules based on confidence-thresholding, including the use of class-specific thresholds (Fumera et al., 2000; Ferri & Hernández-Orallo, 2004) and metric-based threshold optimisation (Shrikumar, Alexandari & Kundaje, 2018). However, as demonstrated in the introduction, confidence scores are not an optimal means of selecting instances for rejection in general, thus leading to the development of alternative, but less interpretable, uncertainty measures (Jiang, Kim, Guan & Gupta, 2018).

There has been work on alternative uncertainty measures to replace confidence scores in threshold-based rejection. For ensembles, classifier votes and linear combinations of confidence scores have been used as uncertainty measures (Rückert & Kramer, 2004; Fumera & Roli, 2004). Other measures have only been developed for certain classification algorithms, such as the disbelief index for SVMs (Wiener & El-Yaniv, 2011), or the variance of confidence scores over Monte-Carlo dropout (MC-dropout) samples in deep neural networks (DNNs) (Geifman & El-Yaniv, 2017). This has been further developed by using a two-dimensional view of uncertainty based on both confidence scores and MC-dropout variance (Yildirim, Ozer & Davulcu, 2019), which could potentially be applied to classification settings other than DNNs by using an ensemble to produce variance estimates. Another alternative to confidence is the trust score, which is based on the distances from an instance to the high-density sets of the closest alternative classes (Jiang et al., 2018).

The general issue presented in the introduction of confidence scores not being aligned with the optimal regions to reject has been noted previously (Cortes et al., 2016). This has led to methods that directly learn a rejection function during classifier training to model relationships between input features and noise, where the function can be

interpreted to understand those relationships (Thulasidasan et al., 2019). This research has primarily involved custom classifiers and loss functions (Cortes et al., 2016; Ni et al., 2019; Shah & Manwani, 2019; Thulasidasan et al., 2019) that allow the classifier to reject (at a user-configured cost) instead of selecting a class. A major drawback to these approaches is that they are model-specific algorithms that require custom implementations, whereas the proposed null-labelling approach is model-agnostic. Another approach that requires minimal algorithm modification is the data duplication method (Sousa & Cardoso, 2013), which achieves rejection through the application of a binary classifier to instances replicated within a higher-dimensional space. By assigning different misclassification costs to instances from each replica dataset, learning a single decision boundary that passes through both replica datasets effectively creates two parallel decision boundaries. The instances classified differently by the two boundaries are rejected. Because this approach only rejects instances near the class boundary, it can be expected to suffer from the same inability to learn regions of class noise that has been demonstrated in confidence-thresholding. Additionally, the multi-class extension of the data duplication method is only designed for ordinal classification problems, as it modifies an ordinal K-class classification method based on $K - 1$ binary classifiers between adjacent classes.

In summary, this literature review has highlighted the distinction between label noise and the more general category of class noise, and it has established why the potential for class noise to result in irreducible errors necessitates rejection methods. Much research on rejection has involved confidence-thresholding. While this method is simple to implement and model-agnostic, it is not an optimal choice for all cases of class noise. The limitations of confidence scores are discussed further in Section 4.3. Another significant line of research has addressed these limitations with custom classifiers that directly learn a rejection function of the input features. However, progress in this research direction is limited by the need to design and implement model-specific methods.

The null-labelling method attempts to bridge the gap between these approaches, as it is a model-agnostic meta-algorithm that allows the region of rejection to be learned directly from input features while enabling model-interpretation methods of the base classifier to be applied to the rejected region.

4.3 Foundations of Confidence-Thresholding Rejection

The goal within the standard classification context is to learn a classifier function $f : \mathcal{X} \rightarrow \mathcal{Y}$ for a given M-dimensional input space $\mathcal{X} = \mathbb{R}^M$ and a set of K classes $\mathcal{Y} = \{c_1, c_2, \ldots, c_K\}$. This is achieved through supervised learning on a training dataset of N instances represented as input/output pairs: $\{X, Y\} = \{(x_i, y_i) | x_i \in \mathcal{X}, y_i \in \mathcal{Y}, i = 1, \ldots, N\}$. This allows a predicted class $\hat{y} \in \mathcal{Y}$ to be produced for any input instance $x \in \mathcal{X}$ as $\hat{y} = f(x)$.

In the context of classification with rejection, the set of possible predictions is extended to include a rejection option $\perp$: $\hat{y} \in \mathcal{Y}', \mathcal{Y}' = \mathcal{Y} \cup \{\perp\}$. The performance of a rejecting-classifier is typically evaluated in terms of its *conditional error rate* (the error rate on non-rejected/covered instances: $E^c = P(\hat{y} \neq y | \hat{y} \neq \perp)$) and *coverage rate* (the proportion of instances not rejected: $C = P(\hat{y} \neq \perp)$), which are typically evaluated empirically on a test dataset. Coverage can also be expressed as the *rejection rate*, which is the proportion of instances that are rejected: $R = 1 - C = P(\hat{y} = \perp)$. A rejecting-classifier should aim to achieve a low conditional error rate while also retaining a high coverage rate.

In order to achieve rejection through confidence-thresholding, the trained classifier is assumed to produce a confidence score for any prediction, represented by the function $g : \mathcal{X} \rightarrow \mathbb{R}$. For a given confidence threshold t, the rejecting-classifier function can be defined as:

$$f_t(x) = \begin{cases} f(x) & \text{if } g(x) \geq t \\ \perp & \text{otherwise} \end{cases} \tag{4.1}$$

Note that users can achieve any coverage rate by selecting t such that the desired proportion of confidence scores achieved for their test dataset have values greater than t. As the likelihood of an instance being misclassified should decrease with a higher confidence score, increasing t should decrease the conditional error rate while also decreasing the coverage rate. However, this entirely depends on the strength of the relationship between confidence scores and the probability of misclassification. In the case of not-at-random class noise, it is important for low confidence scores to be correlated with noise.

Many common classification algorithms (including decision trees and forests, SVMs, logistic regression, and neural networks) learn decision boundaries based on features that are the most discriminative between the target classes (i.e. signal features). This is an optimal strategy when all instances will be classified, but not necessarily when the classifier is allowed to reject some instances. In particular, a rejecting-classifier should also take into account the noise features that discriminate between regions that can be reliably classified and those that cannot. Because linear models like logistic regression base confidence scores on the distances between instances and the decision boundary, a decision boundary not accounting for noise features will result in confidence scores that do not reflect the true reliability of classification. This is demonstrated by the motivating example in Fig. 4.1, where decision boundaries based solely on signal features lead to poor confidence-thresholding performance. However, a sufficiently powerful classifier trained with an objective function to differentiate classes may still be able to learn the structure of regions in the input space that cannot be accurately classified, in which case confidence-thresholding may provide acceptable performance.

Other classification algorithms do not suffer as greatly from this issue because their confidence scores do reflect the neighbourhood of an instance, such as nearest neighbour classifiers (where confidence is based on the agreement between neighbours) and decision trees (where confidence is based on the uniformity within a leaf node). However, the confidence scores of these models can also be unsuitable for confidence-thresholding in some circumstances. Nearest neighbour confidence scores are often required to be weighted by neighbour distances in order to provide adequate granularity for thresholding, and they must be based on an adequate number of neighbours to be representative of the local neighbourhood's uniformity. Shallow decision trees also lack granularity in confidence scores, and large leaf sizes result in confidence scores no longer representative of a small/local neighbourhood. On the other hand, the leaf nodes of a very deep decision tree will be too small to be representative of a neighbourhood.

For the remainder of this chapter, confidence-thresholding and null-labelling will be compared using a logistic regression classifier. Logistic regression is a useful model that is commonly used in medical applications that can often benefit from judicious rejection (e.g. so that difficult cases can be manually reviewed or classified by a more costly test). Logistic regression is also valued for the interpretability of its class coefficients, which can be used to identify noise features through the application of null-labelling.

4.4 Proposed Null-Labelling Method for Rejection

This section presents the novel null-labelling method for iteratively training rejecting-classifiers. In each iteration of null-labelling, a classifier is trained on a modified version of the training dataset with a `null` class assigned to instances that are misclassified by

the previous iteration's classifier. Defining the initial set of training labels as $Y^0 = Y$, then for all null-labelling iterations $j > 0$:

$$y_i^j = \begin{cases} \perp & \text{if } f^{j-1}(x_i) \neq y_i^{j-1} \\ y_i^{j-1} & \text{otherwise} \end{cases} \tag{4.2}$$

where f^j represents the classifier trained on labels Y^j. Note that, for the observed misclassifications to be representative of the classifier's true performance, the training dataset must be classified via k-fold cross-validation. Once a classifier has been trained using a null-labelled training dataset as defined by Equation (4.2), it will be able to directly predict `null` (i.e. reject) just as it can predict any other class value.

Other rejection methods that enable classifiers to reject instances directly require custom loss functions and implementations (Cortes et al., 2016; Thulasidasan et al., 2019), while null-labelling can be applied to any base classifier by simply modifying the training dataset. Null-labelling's creation of the `null` class for noisy instances is also distinct from other label noise mitigation methods that alter the training dataset (Luengo et al., 2018), as they only seek to remove or correct the class of noisy instances.

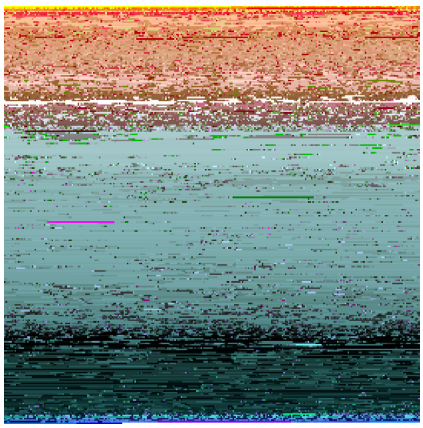
Defining a `null` class enables the classifier to directly learn a rejection function based empirically on its own misclassifications. Regions of noise that cannot be accurately classified will contain many misclassifications and will therefore be relabelled to contain many `null` labels. Regions with few misclassifications will only be sparsely null-labelled, still allowing the classifier to learn the majority classes for those regions. Modelling the regions of noise with an explicit class enables the classifier to take advantage of the *noise features* that define these regions. Furthermore, native interpretation methods for the base classifier can be used to identify the noise features that define the `null` class, as demonstrated in Section 4.6. Additionally, once null-labelling of the regions of noise has been performed, the remaining regions will have higher class

uniformity and be simpler to learn, leading to more reliable decision boundaries for the actual classes. Directly relating features and misclassifications addresses the issues with confidence-thresholding discussed in Section 4.3.

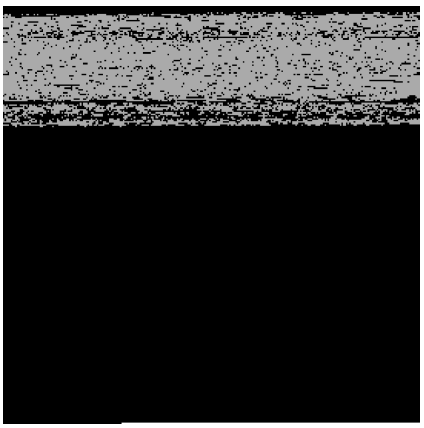
Fig. 4.3 compares the performance of confidence-thresholding (CT) and null-labelling after two iterations (NL-2) on the UCI Skin Segmentation Dataset (Dua & Graff, 2017), with pixels ordered by pre-rejection logistic regression activations. The goal of this dataset is to predict whether or not a given pixel is from an area of skin based only on RGB colour values, and it is reasonable to expect that some colours could commonly appear both on skin and elsewhere in an image. A judicious classifier should reject pixels of such colours. Fig. 4.3 shows that the classifier using null-labelling can better identify which colours to reject, achieving a 70% reduction in conditional error over confidence-thresholding when both classifiers only reject approximately 10% of instances. Not only does null-labelling reject the regions that are still misclassified under confidence-thresholding, but regions that were rejected by confidence-thresholding (because they were nearer the decision boundary) are no longer rejected under null-labelling.

Algorithm 4.1 provides a complete description for implementing null-labelling, including mechanisms for controlling the number of iterations, the achieved coverage rate, and model stability, which are presented in the remainder of this section.

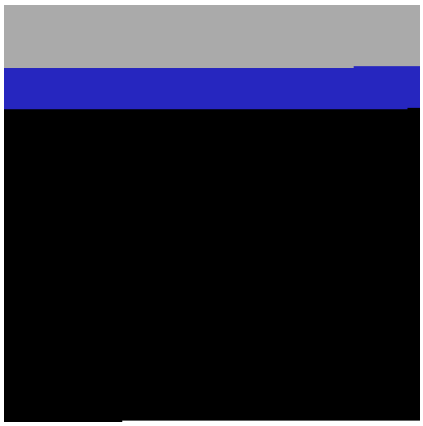
Selecting how many null-labelling iterations to perform depends on the properties that are desired of the final classifier. Because null-labelling is a greedy process (instances assigned the `null` class in a previous iteration cannot be reverted to their original class), the coverage rate will tend to decrease with subsequent iterations, typically with a corresponding decrease in conditional error. Therefore, if the goal is to achieve the smallest possible conditional error for a minimum allowable coverage rate (i.e. bounded rejection), iterations can be stopped once the classifier falls below the minimum coverage rate (evaluated on a validation dataset separate from the training



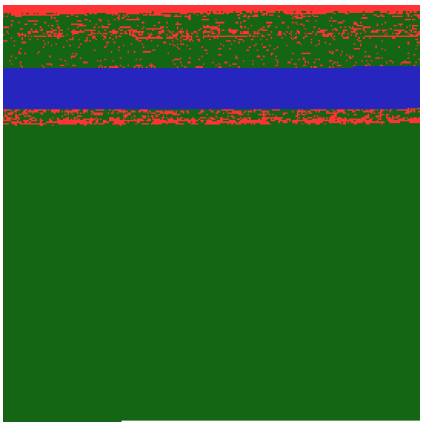
(b) Test dataset pixel colours



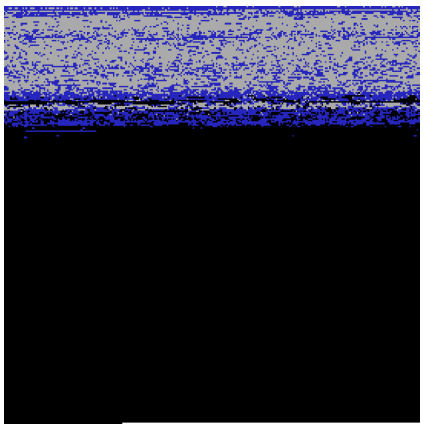
(c) Class labels (20.75% `skin`)



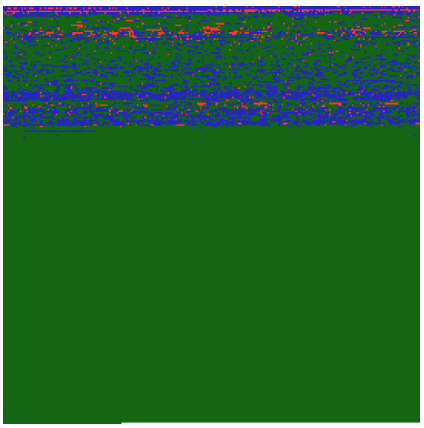
(d) CT classifications
(10.00% rejected)



(e) CT errors
(6.15% conditional error)



(f) NL-2 classifications
(10.80% rejected)



(g) NL-2 errors
(1.60% conditional error)

Figure 4.3: Comparison of CT and NL-2 on the skin segmentation dataset.

Algorithm 4.1: Application of null-labelling with stopping criteria and coverage controls.

Input : Training dataset $\{X, Y\}$, Validation dataset $\{X^v, Y^v\}$, Max iterations J, NL rate θ, CV folds k, CV repetitions H, Min CV repetition consensus η, OPTIONAL Min coverage $C_{\min}$ OR Max conditional error $E^c_{\max}$ OR Rejection cost λ_r

Output : A set of potential rejecting-classifiers

$Y^0 \leftarrow Y$ **for** $j \leftarrow 0$ **to** J **do**
  Train classifier f^j on $\{X, Y^j\}$;
  Evaluate f^j on $\{X^v, Y^v\}$ to determine C_j and E^c_j;
  if $C_{\min}$ *provided and* $C_j < C_{\min}$ **then**
    // Early return for bounded rejection;
    return $\{f^{j-1}\}$;
  end
  if $E^c_{\max}$ *provided and* $E^c_j \leq E^c_{\max}$ **then**
    // Early return for bounded error;
    return $\{f^j\}$;
  end
  for k*-fold CV runs* $h \leftarrow 0$ **to** H *over* $\{X, Y^j\}$ **do**
    Train classifier $f^{j,h}$ and confidence-scorer $g^{j,h}$ to cover the domain of X;
  end
  Construct Y^{j+1} according to Equation (4.4);
end
if λ_r *given* **then**
  // Minimise cost as defined in Equation (4.3);
  return $\{f^j | j \in \arg\min_{j \in 0,\ldots,J} \mathbb{E}[\mathcal{C}_j]\}$
end
// No selection criteria met, so return all classifiers;
return $\{f^0, \ldots, f^J\}$;

dataset), as in lines 4–6 of Algorithm 4.1. Conversely, if the goal is to achieve as much coverage as possible for a maximum conditional error rate (i.e. bounded error), then iterations can be stopped once the classifier achieves the desired conditional error rate on a validation dataset, as in lines 7–9 of Algorithm 4.1. The maximum number of iterations J can be set as a hard limit on the computation to handle cases where such constraints are never exceeded. If the above constraints are not given for the application, then J should be set high enough to achieve a broad range of coverage rates. Users can then inspect a tradeoff plot like Fig. 4.1d to select their preferred classifier. Alternatively, if the cost of instance rejection (λ_r) can be specified in the range $(0, 1)$ (where 0 denotes the cost of a correct classification and 1 denotes the cost of a misclassification), then the cost-optimal classifier can be found by selecting the classifier with minimal expected cost $\mathcal{C}$ as defined in Equation (4.3) (from Hanczar, 2019) and applied in lines 15–17 of Algorithm 4.1.

$$\mathbb{E}[\mathcal{C}] = CE^c + (1 - C)\lambda_r \tag{4.3}$$

If the base classifier is unstable in relation to training set changes, then repeated k-fold cross-validation (CV) can be used to decide which instances to null-label, similar to the label-noise filtering strategy of Brodley and Friedl (1999). The k-fold cross-validation used to identify misclassified instances can be repeated H times (lines 10–12 of Algorithm 4.1) with different random splits. This will produce a set of classification results for each training instance, and the set of null-labelled instances can be limited to those that would be null-labelled by a minimum consensus (η) of repetitions, which should be set to represent at least a majority consensus ($\frac{\eta}{H} > 0.5$). If classifier stability is not of concern, then default values of $H = \eta = 1$ can be used for efficiency.

Given the above parameters, the complexity of null-labelling can be expressed as $O(JHk)$ invocations of the underlying base classifier for both training and testing.

This complexity is derived from the iterations over J and H on lines 1 and 10 in Algorithm 4.1, as well as the k-folds in each application of cross-validation. The typical ranges for each of these parameters are relatively small. Initial experimentation found $J = 10$ is often more than sufficient to achieve an adequate range of coverage rates. H can be set to 1 except when the classifier is unstable to training set changes, in which case $H \leq 10$ is reasonable. Finally, experiments in Section 4.6 show that $k = 5$ is sufficient for identifying noisy instances via cross-validation.

One challenge to effectively applying null-labelling lies in achieving a desired coverage rate. If the error rate of the original classifier is very low, then many instances may be null-labelled in even the first iteration, resulting in an initially low level of coverage that only continues to decrease with subsequent iterations. In order to provide more control over the level of coverage achieved by null-labelling, the set of null-labelled instances can be reduced to only include instances misclassified with low confidence scores. This technique essentially uses confidence-thresholding to contribute to the decision of which instances to null-label. Its influence can be controlled by a null-labelling rate θ that determines the proportion of misclassified instances that will be null-labelled. Given that θ introduces dependence on the quality of confidence scores, it should be set to a default value of 1 and only reduced when necessary to increase coverage.

The θ, H, and η parameters can be incorporated in the following reformulation of Equation (4.2) (applied on line 13 of Algorithm 4.1) to construct the training set for each null-labelling iteration:

$$\Xi^{j,h} = L\big(g^{j,h}, \theta, \{x_i | f^{j,h}(x_i) \neq y_i^j\}\big)$$

$$y_i^j = \begin{cases} \perp & \text{if } \big|\{h | x_i \in \Xi^{j-1,h}\}\big| \geq \eta, h \in 1, \ldots, H \\ y_i^{j-1} & \text{otherwise} \end{cases} \tag{4.4}$$

where $f^{j,h}$ and $g^{j,h}$ are the classifier and confidence-scoring functions for CV repetition h of iteration j, and $L(r, q, S)$ selects the lowest proportion q of elements in set S as ranked by function r. $\Xi^{j,h}$ represents the proportion θ of instances classified with the lowest confidence scores that were misclassified by CV repetition h of iteration j, which are then aggregated over H iterations so that only instances that appear at least η times are null-labelled.

4.5 Combining Sets of Rejecting-Classifiers

While the threshold used in confidence-thresholding can be varied to achieve a given coverage rate, the classifier produced by each null-labelling iteration has a fixed coverage rate. This section presents a framework for combining a given set of rejecting-classifiers into a composite classifier to achieve a desired coverage rate. Such composite classifiers are inspired by the concept of "proportionally mixed classifiers" proposed by Ferri and Hernández-Orallo (2004).

Two rejecting-classifiers A and B can be combined to produce a composite classifier D by randomly selecting which classifier to apply with probability $P(A) = 1 - P(B) = p$. In order to evaluate this composite classifier, its performance metrics can be expressed in terms of metrics for the original classifiers A and B. The conditional error rate E^c used so far is intuitive to understand in tradeoff plots (such as Fig. 4.1d) as the error rate over covered instances. It turns out that the conditional error rate for a composite classifier is a function of the unconditional error and coverage rates for the original classifiers, as shown in Theorem 4.1. The unconditional error rate represents the error rate achieved over all classified instances, including covered and rejected instances: $E^u = P(\hat{y} \notin \{y, \perp\})$. Therefore, the unconditional error rate is the product of the conditional error rate and coverage rate, as shown in Equation (4.5). Consequently, when Algorithm 4.1 minimises the product of conditional error and coverage contained

in the classifier cost (Equation (4.3)), it is effectively minimising the unconditional error.

$$\begin{aligned} E^c \times C &= P(\hat{y} \neq y | \hat{y} \neq \perp) \times P(\hat{y} \neq \perp) \\ &= P(\hat{y} \notin \{y, \perp\}) = E^u \end{aligned} \tag{4.5}$$

Theorem 4.1. *The expected conditional error rate ($\mathbb{E}[E^c_D]$) of composite classifier D is given by $\frac{pE^u_A + (1-p)E^u_B}{pC_A + (1-p)C_B}$, where E^u_A, E^u_B, C_A, and C_B are the unconditional error rates and coverage rates for original classifiers A and B, and p is the probability of applying classifier A.*

Proof. Intermediate results are first proved for the expected unconditional error $\mathbb{E}[E^u_D]$ and coverage $\mathbb{E}[C_D]$ rates:

$$\begin{aligned}
\mathbb{E}[E^u_D] &= P(\hat{y} \notin \{y, \perp\}) \\
&= P(A)P(\hat{y} \notin \{y, \perp\}|A) + P(B)P(\hat{y} \notin \{y, \perp\}|B) \\
&= pE^u_A + (1-p)E^u_B \\
\mathbb{E}[C_D] &= P(\hat{y} \neq \perp) \\
&= P(A)P(\hat{y} \neq \perp|A) + P(B)P(\hat{y} \neq \perp|B) \\
&= pC_A + (1-p)C_B \\
\mathbb{E}[E^c_D] &= P(\hat{y} \neq y | \hat{y} \neq \perp) \\
&= P(A|\hat{y} \neq \perp)P(\hat{y} \neq y|\hat{y} \neq \perp, A) + P(B|\hat{y} \neq \perp)P(\hat{y} \neq y|\hat{y} \neq \perp, B) \\
&= \frac{P(A) \times P(\hat{y} \neq \perp|A)}{P(\hat{y} \neq \perp)} \times E^c_A + \frac{P(B) \times P(\hat{y} \neq \perp|B)}{P(\hat{y} \neq \perp)} \times E^c_B \\
&= \frac{pC_A E^c_A}{\mathbb{E}[C_D]} + \frac{(1-p)C_B E^c_B}{\mathbb{E}[C_D]} \\
&= \frac{pE^u_A + (1-p)E^u_B}{pC_A + (1-p)C_B}
\end{aligned}$$

□

The first part of Theorem 4.1 shows that the composite classifier's expected unconditional error rate ($\mathbb{E}[E^u_D]$) is a linear combination of the unconditional error rates for classifiers A and B and that the same property also holds for the expected coverage rate $\mathbb{E}[C_D]$. This is consistent with the claim of Ferri and Hernández-Orallo (2004) that "proportionally mixed classifiers" can be represented as a linear interpolation between the original two classifiers on a plot of unconditional error against coverage[2].

Furthermore, the second part of Theorem 4.1 proves that the composite classifier's expected conditional error rate ($\mathbb{E}[E^c_D]$) can be expressed in terms of the unconditional error and coverage rates of classifiers A and B, mirroring the relationship in Equation (4.5) ($E^c = \frac{E^u}{C}$). Theorem 4.2 also proves that this simple formulation for $\mathbb{E}[E^c_D]$ is equivalent to the non-linear interpolation scheme for conditional error proposed by Hanczar (2019). Note that while Hanczar's notation interpolates between classifiers $\mathbf{X}$ and $\mathbf{0}$ to produce a classifier $\mathbf{0} + \mathbf{x}$, the notation used here continues to refer to these classifiers as A, B, and D, respectively. Based on their notation, classifier A rejects X more instances than B and D rejects $x = pX$ more instances than B. The following notation is also used in this chapter:

- G and M are the counts of instances rejected by classifier A that are correctly and incorrectly classified by classifier B ($G = X - M$).
- N is the total number of instances.
- $r = NR = N(1 - C)$ is the number of instances rejected by a classifier.
- $R_\Delta = R_A - R_B = C_B - C_A = \frac{X}{N}$

Finally, the corrected definition of $M = N(E^u_B - E^u_A)$ must be used in place of $M = N(E^c_B - E^c_A)$ (as stated by Hanczar), because the E^c values are relative to different coverage rates.

[2]Though the authors refer to"unconditional error" simply as error, and plot the inverse of coverage (rejection), which they refer to as "abstention".

Theorem 4.2. *Hanczar's (2019) method of non-linear interpolation for two classifiers A and B also results in an expected conditional error given by* $\mathbb{E}[E^c_D] = \frac{pE^u_A + (1-p)E^u_B}{pC_A + (1-p)C_B}$.

Proof. Hanczar defines conditional error for interpolated classifier D with g correct (and $m = x - g$ incorrect) classifications out of x extra instances classified as:

$$\begin{aligned} E^{c\{g\}}_D &= E^c_B \frac{N - r_B}{N - r_B - g - m} - \frac{m}{N - r_B - g - m} \\ &= \frac{E^c_B(N - r_B) - (x - g)}{N - r_B - g - (x - g)} \\ &= \frac{E^c_B(N - r_B) - x + g}{N - r_B - x} \end{aligned}$$

Hanczar then sums conditional errors for each possible number of correct classifications g weighted by $P(g)$, which is given by the binomial distribution $\mathcal{B}(x, \frac{G}{X})$:

$$\mathbb{E}[E^c_D] = \sum_{g=0}^{x} P(g) \times E^{c\{g\}}_D$$

Because the conditional error $E^{c\{g\}}_D$ is a linear function of g, its expected value over possible values of g can be expressed as the conditional error for the expected value of g, which is $x\frac{G}{X}$, the mean of $\mathcal{B}(x, \frac{G}{X})$:

$$\begin{aligned} \mathbb{E}[E^c_D] = E^{c\{x\frac{G}{X}\}}_D &= \frac{E^c_B(N - r_B) - x + x\frac{G}{X}}{N - r_B - x} \\ &= \frac{E^c_B(N - NR_B) - pX + pX\frac{X-M}{X}}{N - NR_B - pX} \\ &= \frac{E^c_B(1 - R_B) - pR_\Delta + pR_\Delta\frac{R_\Delta - (E^u_B - E^u_A)}{R_\Delta}}{1 - R_B - pR_\Delta} \\ &= \frac{E^c_B C_B - pR_\Delta + p(R_\Delta - E^u_B + E^u_A)}{C_B - p(C_B - C_A)} \\ &= \frac{E^u_B - pE^u_B + pE^u_A}{C_B - pC_B + pC_A} = \frac{pE^u_A + (1-p)E^u_B}{pC_A + (1-p)C_B} \end{aligned}$$

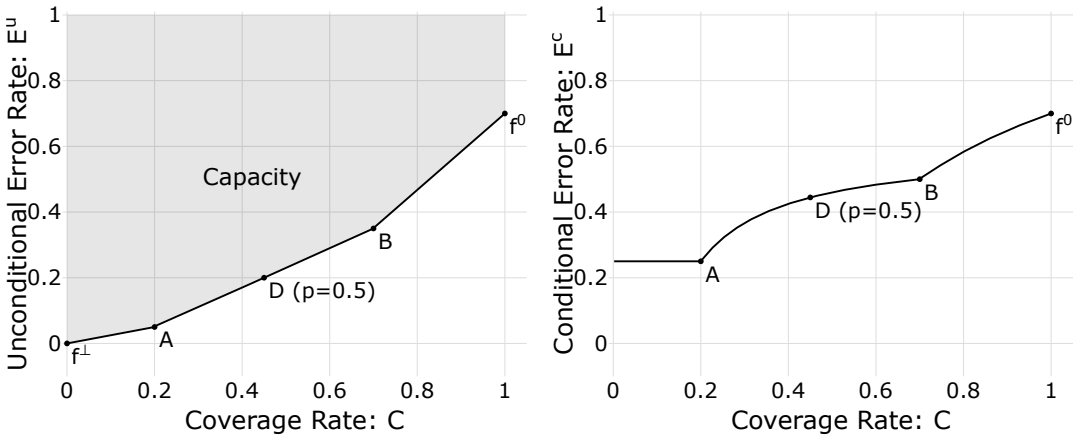
□

This framework for combining rejecting-classifiers also supports using the capacity curve and metric proposed by Ferri and Hernández-Orallo (2004) to compare the performance of sets of rejecting-classifiers. In order to construct a capacity curve for a set of classifiers, each classifier's coverage rate is first plotted against its unconditional error rate. For the full range of coverage rates to be covered, points are also plotted for the non-rejecting base classifier (f^0) and the fully rejecting classifier ($f^\perp(x) = \perp$)[3]. As linear interpolation between any two classifiers represents a composite classifier, the convex hull of classifier points represents a Pareto front where the coverage/error tradeoff of any point above the curve will be dominated by a point on the curve. An example of such a capacity curve is plotted in Fig. 4.4a. Furthermore, the plot of coverage against conditional error in Fig. 4.4b demonstrates the results of the non-linear interpolation required for conditional error.

From an end-user perspective, the conditional error E^c of a classifier should be as small as possible for a given coverage level C. Given that unconditional error decreases monotonically with coverage, it can be concluded from Equation (4.5) that classifiers with a rapid decrease of unconditional error E^u for a decrease in coverage C are preferred over ones whose decline is less rapid. In turn, this implies that the area above the plot of unconditional error versus coverage (i.e. the capacity metric) should be as large as possible. Furthermore, the capacity metric evaluates the performance that can be achieved over the full range of coverage rates $[0, 1]$, similar to how the AUC for ROC and PRC curves evaluates performance over all classification thresholds. Therefore, the capacity metric is used to compare the sets of rejecting-classifiers produced by confidence-thresholding and null-labelling in Section 4.6.

Note that while linear interpolation between classifiers could produce a classifier closer to a bounded rejection or bounded conditional error limit, it cannot produce a

[3] Artificial variations of rejecting-classifiers with full coverage are not considered here as they were by Ferri and Hernández-Orallo, because estimating their performance relies on assumptions of class distributions.



(a) Plotting C vs. E^u (b) Plotting C vs. E^c

Figure 4.4: Plotting and interpolation between classifiers.

classifier with a lower expected cost (as defined in Equation (4.3)). This is because the expected cost of the interpolated classifier D is a linear function of p and the expected costs of A and B. Therefore, $\mathbb{E}[\mathcal{C}_D]$ must be minimised when $p = 1$ or $p = 0$:

$$\begin{aligned}
\mathbb{E}[\mathcal{C}_D] &= C_D E^c_D + (1 - C_D)\lambda_r \\
&= E^u_D + R_D\lambda_r \\
&= (pE^u_A + (1-p)E^u_B) + (pR_A + (1-p)R_B)\lambda_r \\
&= p(E^u_A + R_A\lambda_r) + (1-p)(E^u_B + R_B\lambda_r) \\
&= p\mathbb{E}[\mathcal{C}_A] + (1-p)\mathbb{E}[\mathcal{C}_B] \\
\arg\min_p \mathbb{E}[\mathcal{C}_D] &= \begin{cases} 1 & \mathbb{E}[\mathcal{C}_A] < \mathbb{E}[\mathcal{C}_B] \\ 0 & \mathbb{E}[\mathcal{C}_A] > \mathbb{E}[\mathcal{C}_B] \\ [0,1] & \mathbb{E}[\mathcal{C}_A] = \mathbb{E}[\mathcal{C}_B] \end{cases}
\end{aligned} \tag{4.6}$$

4.6 Experimental Study

The following experimental study demonstrates null-labelling's superior handling of class noise compared to confidence-thresholding. The impact of the correlation between signal and noise on null-labelling and confidence-thresholding is also explored. Furthermore, the coefficients of the *null* class are interpreted to identify features correlated with class noise (so-called "noise features"), and the proposed θ parameter is applied to increase the coverage rate achieved by null-labelling. All source-code for the experimentation (including dataset pre-processing) has been made available online at: `https://github.com/ben-denham/pyrejection`.

4.6.1 Experiment Framework

The scikit-learn implementation of logistic regression is applied using the `lbfgs` solver, a maximum of 10,000 iterations (never reached), and a multinomial loss for multi-class datasets. Except where noted otherwise, null-labelling is performed without repetitions ($H = 1, \eta = 1$) on all misclassifications ($\theta = 1$) from 5-fold cross-validation.

In order to compare confidence-thresholding and null-labelling in the presence of class noise, noise is injected into a set of benchmark datasets from the UCI Machine Learning Repository (Dua & Graff, 2017) listed in Table 4.1. These datasets represent a variety of instance (N), class (K), and feature (M) counts (including numerical and categorical features), and many have been used in past studies on rejection. While the Gaussian distribution is commonly used to model feature noise, these experiments use a distribution that simulates not-at-random class noise. In order to achieve this, a `noise_feature` is added to each dataset with values uniformly distributed in range $[0, 1]$. Instance class values are then re-assigned with probability increasing exponentially with the `noise_feature`: $P(\text{noise}|x) = \lambda e^{-\lambda s(1-x_{\texttt{noise\_feature}})}$. Setting $\lambda = 0.8$ produces an 80% re-assignment rate when $x_{\texttt{noise\_feature}} = 1$. s is used to scale

Table 4.1: Benchmark datasets for classification with rejection.

Dataset	N	K	**Num.** M	**Cat.** M
ARM (AReM - top 5 activities)	35,999	5	6	0
BNK (Banknote)	1372	2	4	0
DIA (Diabetic)	1151	2	16	3
ELE (Electrical)	10,000	2	13	0
EYE	14,980	2	14	0
DIG (Handwritten Digits)	1797	10	64	0
GAS	13,910	6	128	0
LED	10,000	10	0	7
LTR (Letter Recognition)	20,000	26	16	0
MUS (Mushroom)	8124	2	0	22
PHI (Phishing)	11,055	2	0	30
SEG (Statlog - Segment)	2310	7	19	0
VEH (Vehicle)	846	4	18	0

the $[0, 1]$ `noise_feature` values such that they range over 95% of the probability mass of the exponential distribution, and it is computed as the 95% quantile of the distribution: $s = -\frac{\ln(1-0.95)}{\lambda}$. If a class value is re-assigned, it will be replaced with a random selection from all possible class values (which may result in selecting the instance's original class value). This exponential noise distribution simulates a scenario in which the correlation between input features and class values holds for typical feature values but not when the noise feature(s) reach extreme (outlier) values. Note that because of this noise injection, the typical feature noise used with the LED dataset was not included in the generated dataset.

In addition to the benchmark datasets, experiments are performed with synthetic "radial" datasets that vary the correlation between the signal features that determine the class value and noise features that determine the rate of class noise. Null-labelling should outperform confidence-thresholding to a greater extent when the signal and noise features are orthogonal to each other. 2-dimensional (x_1, x_2) datasets are generated with instances randomly distributed within the unit circle and an initial binary class value of 1 when $x_1 \geq 0$ and 0 otherwise (i.e. X_1 is the signal feature). Exponentially

distributed class noise is introduced by randomly re-assigning class values with probability $P(\text{noise}|x) = \lambda e^{-\lambda s(1-\omega_x)}$, using the same values of λ and s as for the benchmark datasets, and where ω_x represents a combination of x_1 and x_2 values that is mapped from range $[-1, 1]$ to $[0, 1]$:

$$\omega_x = \frac{\Big([x_1, x_2] \cdot [\cos(\alpha), \sin(\alpha)]\Big) + 1}{2} \tag{4.7}$$

where α is an angle that determines the correlation between the signal feature x_1 and the probability of noise. Experiments are performed with a range of α values from $0°$ (signal and noise are fully correlated / both determined by x_1) to $90°$ (signal and noise are orthogonal / x_1 is a signal feature, x_2 is a noise feature). These datasets are referred to as `R00` through `R90`.

All datasets are randomly divided into stratified 70/30 train/test splits for performance evaluation, with z-score normalisation applied to numeric features and drop-first one-hot encoding applied to categorical features. Dataset pre-processing is performed before train/test splitting, as the influence of differences in train/test distribution is not part of the evaluation.

Experiments are performed with confidence-thresholding (CT) and null-labelling (NL) on each dataset. Nine iterations of null-labelling are performed on each dataset, allowing a capacity curve to be plotted for classifiers $f_0, \ldots, f_9$. For confidence-thresholding, a capacity curve is created for each dataset using ten thresholds that achieve coverage rates $[0.1, 0.2, \ldots, 1]$, matching the number of rejecting-classifiers produced with null-labelling. These capacity curves are used to determine the capacity metric, as well as the unconditional error rate achieved for a coverage rate of 80% ($E^u\%$ @ 80% C) and the coverage rate achieved for a 50% reduction in unconditional error rate as compared to the classifier with 100% coverage ($C\%$ @ 50% of E^u).

4.6.2 Experiment Results

One hundred experiments were performed on each dataset with different randomly selected train/test splits, resulting in one hundred capacity curves and associated metrics for each rejection method on each dataset. The metric means are presented in Table 4.2, along with the maximum standard deviation over both methods (for all metrics and all datasets other than MUS, NL achieved a lower variance). Corrected re-sampled t-tests were performed to test the null hypothesis that "there was no significant difference between the metric values achieved with confidence-thresholding and null-labelling". The superior method is bold-faced in Table 4.2 when the null hypothesis was rejected with $p < 0.05$ for that dataset and metric. These results were cross-checked with a two-sided Wilcoxon signed-rank test, which rejected the null hypothesis with $p < 0.05$ for all datasets and metrics.

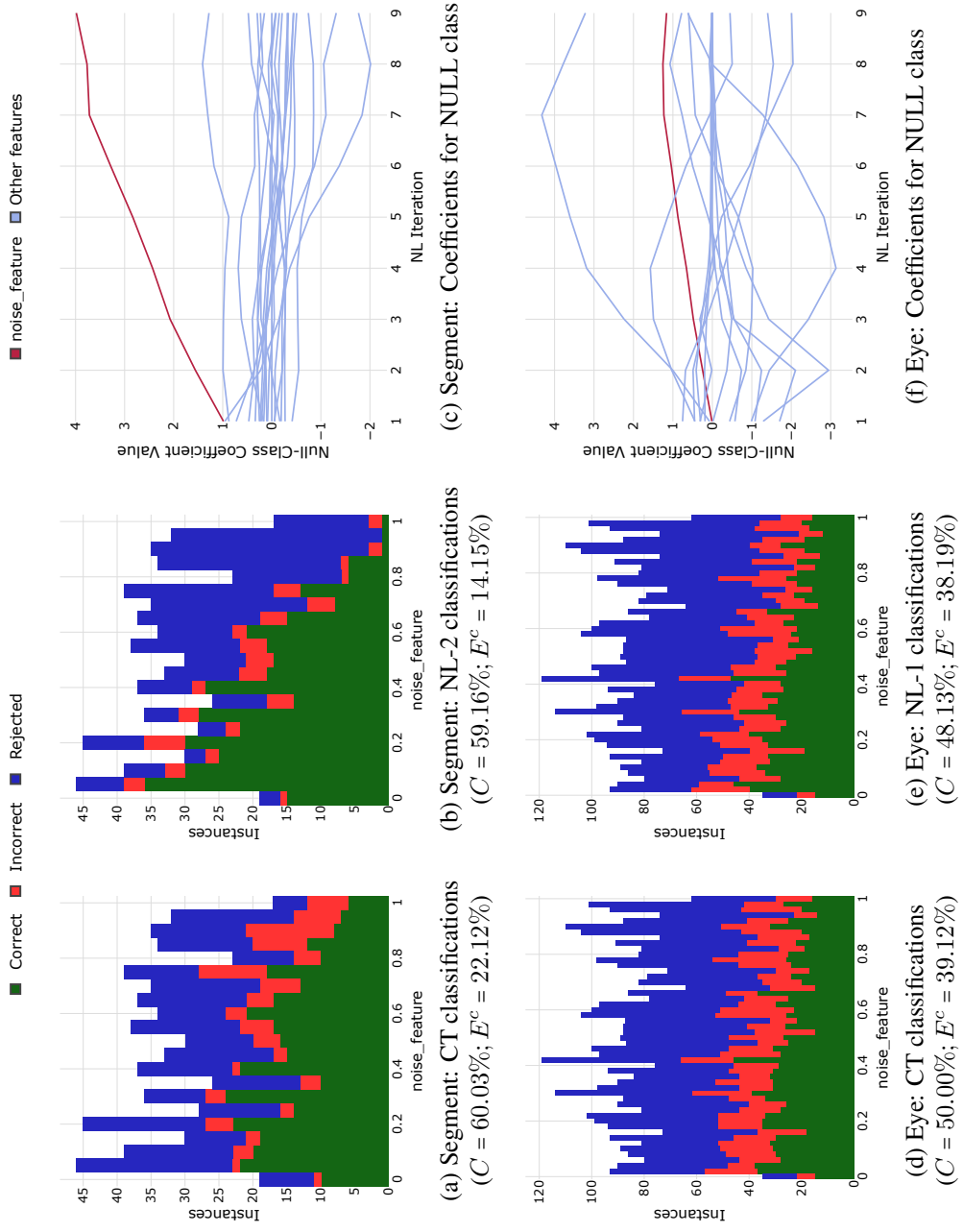
Table 4.2: Results for CT and NL on noisy datasets.

	Capacity			E^u**% @ 80%** C			C**% @ 50% of** E^u		
	CT	**NL**	σ	**CT**	**NL**	σ	**CT**	**NL**	σ
ARM	.800	**.805**	.002	**33.3**	34.2	0.3	58.2	58.4	0.5
BNK	.941	**.951**	.007	9.6	**8.2**	1.1	59.5	**73.1**	3.7
DIA	**.859**	.844	.011	**24.9**	26.4	1.8	**61.8**	56.7	2.8
ELE	.937	**.953**	.002	10.1	**8.4**	0.4	59.4	**75.0**	1.7
EYE	.796	**.797**	.004	34.1	34.2	0.5	54.3	54.5	0.6
DIG	.872	**.889**	.009	21.0	20.7	1.4	64.2	**70.4**	2.3
GAS	.891	**.927**	.003	17.5	**13.5**	0.5	54.7	**75.0**	1.2
LED	.893	**.933**	.003	17.7	**12.6**	0.5	54.0	**77.4**	1.5
LTR	.821	**.829**	.002	**31.1**	32.5	0.4	**65.1**	63.9	0.4
MUS	.937	**.951**	.004	10.2	**7.7**	0.6	50.9	**70.6**	4.6
PHI	.933	**.942**	.003	10.7	10.5	0.5	68.0	**72.9**	1.8
SEG	.887	**.908**	.007	19.4	**17.8**	1.2	66.5	**74.4**	2.2
R90	.946	**.957**	.006	8.7	**7.3**	1.0	57.8	**72.2**	4.0
R65	.947	**.956**	.006	8.4	**7.4**	1.0	62.7	**73.8**	4.2
R45	.944	**.954**	.005	9.0	**8.1**	0.9	66.5	**75.5**	4.6
R25	.943	**.950**	.006	9.0	8.6	1.0	72.7	**77.0**	4.7
R00	.942	**.951**	.006	9.0	8.9	1.0	78.1	79.9	3.8

Null-labelling significantly outperformed confidence-thresholding on all three metrics in Table 4.2. Null-labelling achieved greater capacity than confidence-thresholding for all datasets except DIA and was also able to achieve lower error and higher coverage at the fixed points more often than confidence-thresholding. The results with the radial datasets also confirm the hypothesis that improvement of null-labelling over confidence-thresholding tends to increase with the cosine of the angle between the signal and noise features, reaching a maximum at $90°$.

Fig. 4.5 presents an in-depth examination of two datasets to provide greater insight into the performance of null-labelling in relation to confidence-thresholding. For the Segment dataset, Fig. 4.5b shows that NL captures the relationship between the `noise_feature` and the misclassification rate: rejecting more instances with high `noise_feature` values and fewer instances with low `noise_feature` values. In Fig. 4.5a, CT fails to capture this relationship as the rate of rejection is approximately uniform across the range of `noise_feature` values, thus exposing the limitation of confidence as a measure for estimating noise. NL's superior noise detection capability explains why it can achieve a lower conditional error rate at approximately the same coverage rate as CT. Conversely, the plots in Figs. 4.5d and 4.5e show a much higher error rate with little correlation to the `noise_feature` for the Eye dataset, indicating that other factors have a greater impact on classification accuracy than the injected class noise. These inherent misclassifications do not appear to be strongly correlated with any particular "noise" features, resulting in rejection performance from NL that is not much better than that of CT. Similar results were observed with the Diabetic dataset, for which NL does not perform as well as CT.

Furthermore, Figs. 4.5c and 4.5f demonstrate how the coefficients learned for null-labelling's `null` class can be interpreted to identify which features are correlated with misclassifications/class noise. For the Segment dataset, the coefficient for the



(a) Segment: CT classifications (C = 60.03%; E^c = 22.12%)

(b) Segment: NL-2 classifications (C = 59.16%; E^c = 14.15%)

(c) Segment: Coefficients for NULL class

(d) Eye: CT classifications (C = 50.00%; E^c = 39.12%)

(e) Eye: NL-1 classifications (C = 48.13%; E^c = 38.19%)

(f) Eye: Coefficients for NULL class

Figure 4.5: Comparison of CT and NL on benchmark datasets.

`noise_feature` tends to increase with subsequent null-labelling iterations, indicating that instances with high `noise_feature` values will be assigned the `null` class. Therefore, the `noise_feature` can be interpreted as strongly correlated with the class noise. In the case of the Eye dataset, the `noise_feature` coefficient also increases with subsequent null-labelling iterations. However, other features with coefficients of greater magnitude indicate that the `noise_feature` is not correlated with the majority of inherent misclassifications in the dataset. Note that the coefficients for the `noise_feature` may not always increase monotonically; when each coefficient is re-learned by logistic regression at each iteration, regularisation results in coefficients that are only sufficient for class discrimination relative to all other coefficients.

Finally, Fig. 4.6 demonstrates the use of θ to more precisely control the coverage rate achieved by null-labelling on the letter-recognition dataset. Decreasing the value of

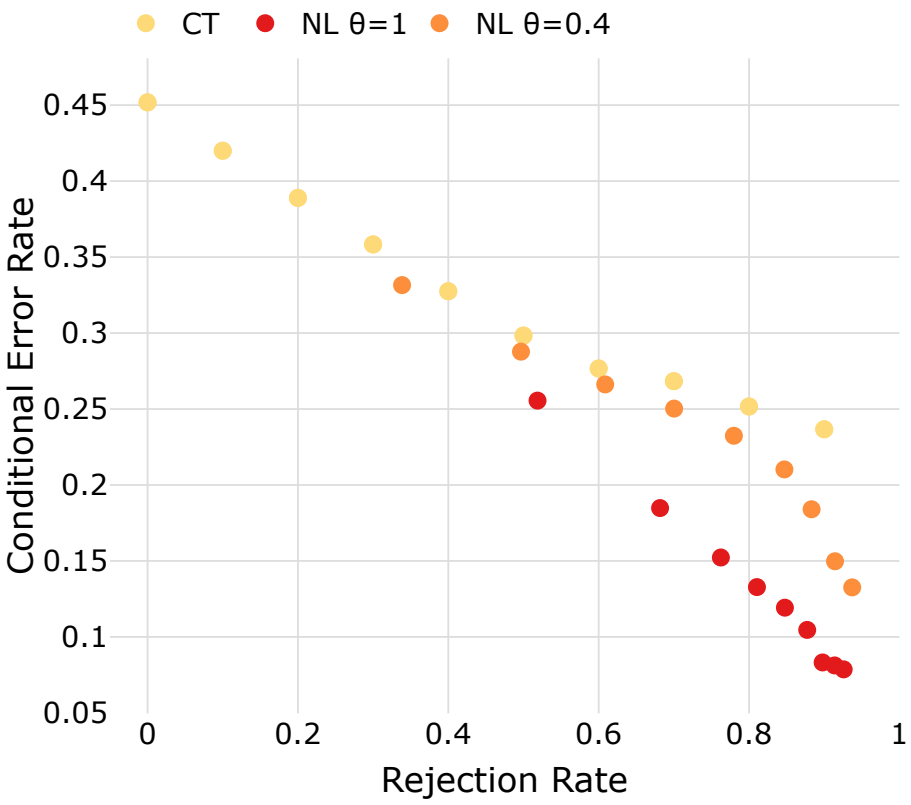


Figure 4.6: Increasing NL coverage on letter-recognition.

θ to 0.4 achieves a lower rejection/higher coverage rate. However, this comes at the cost of conditional error rates closer to those achieved with confidence-thresholding because the same confidence scores are used to influence the selection of instances to null-label.

4.7 Conclusion

This chapter has demonstrated shortcomings of rejection by confidence-thresholding in the presence of not-at-random class noise and addressed RO3 by proposing an alternative rejection method: null-labelling. Through iterative learning from the empirical misclassifications of a model, null-labelling enables a classifier to learn the relationships between input features and class noise directly. Experiments have demonstrated null-labelling's capability to provide a better tradeoff between coverage and error using an evaluation based on a framework for combining sets of rejecting-classifiers that unifies prior theories. Experiments have also demonstrated how a classification model produced by null-labelling can be interpreted to identify features correlated with class noise.

Chapter 5

Integration of Components for a Tolerant Machine Learning Architecture

5.1 Introduction

To clearly illustrate how the contributions presented in earlier chapters can be applied together, this chapter presents an architecture for an end-to-end tolerant machine learning system. The presentation of the architecture is based on the standard 4+1 architectural view model (Kruchten, 1995). Specifically, a logical view of the architecture is presented to highlight the architecture's reliability and modifiability, followed by a set of usage scenarios that demonstrate the flexibility of the architecture.

The context diagram in Fig. 5.1 presents the operating environment for a tolerant machine learning system, where its primary role is to classify and quantify incoming unlabelled data from some data source. Users of the system are categorised into two primary roles. Firstly, the data consumers receive the classification and quantification outputs and set the requirements for output quality. Secondly, the domain experts are

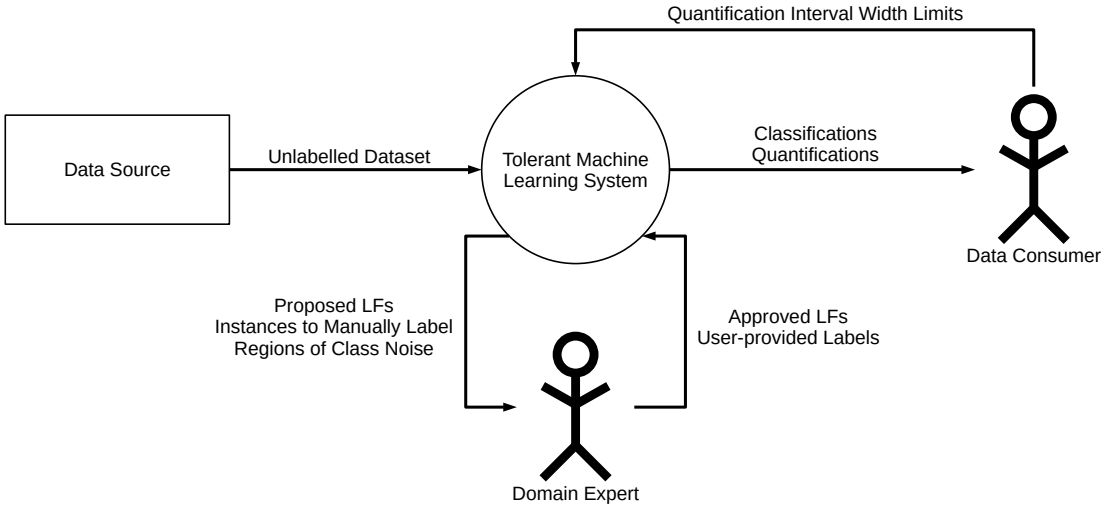


Figure 5.1: Context diagram for the tolerant machine learning system.

responsible for training the system. The elements representing inputs and outputs to the system are described further in the logical view (Section 5.2).

The design decisions that produced the architecture were guided by the following primary qualities desired for a tolerant machine learning system:

- **Reliability** - In order for data consumers to derive value from a machine learning system, they must be able to trust the validity of its outputs in all circumstances. Therefore, the system must be able to consistently produce outputs that are accurate enough to meet the requirements of data consumers despite any training data deficiencies. The contributions presented in previous chapters are the primary mechanism for delivering this reliability.
- **Modifiability** - For the investment required to deploy a machine learning system to be worthwhile, it must be able to provide meaningful value over a prolonged lifetime. In order to keep up with the constantly evolving state-of-the-art in machine learning, the system should be composed of independently replaceable, modular components with clear responsibilities. This will enable individual components to be improved with minimal impact on other components. The

variability guide in Section 5.2.3 highlights the modifiability of the architecture.

- **Flexibility** - In order to support users with different pre-existing data and knowledge inputs for training (e.g. manually labelled instances, labelling functions, or nothing at all) and desired result formats (e.g. classifications or quantifications), the system should support a variety of interaction modes. The set of usage scenarios presented in Section 5.3 demonstrate the architecture's flexibility.

5.2 Logical View of the Architecture

The following logical view documents the components in the architecture, the relationships and interactions between them, and the data artefacts they consume and generate. A logical view was selected for documenting the architecture to identify the reliability-supporting components that map to contributions in previous chapters and to showcase the modularity of the architecture that supports its modifiability.

The primary presentation of this view is divided into two diagrams, with a legend for the diagram format provided in Fig. 5.2. Note that a distinct visual representation is used to highlight data artefacts that provide details for individual data instances, such as instance features or class labels. Fig 5.3 presents the components used to train classifiers, while Fig 5.4 presents the components used to apply classifiers and produce quantifications. Components with contributions from this thesis are highlighted in bold. The following subsections describe the elements within each diagram and highlight points of variability in the architecture. An application of classifying and quantifying faults in textual descriptions of warranty claims is used as a running example.

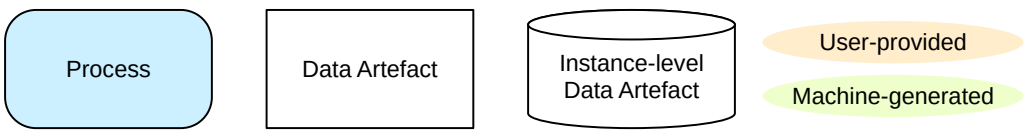


Figure 5.2: Legend for logical view diagrams.

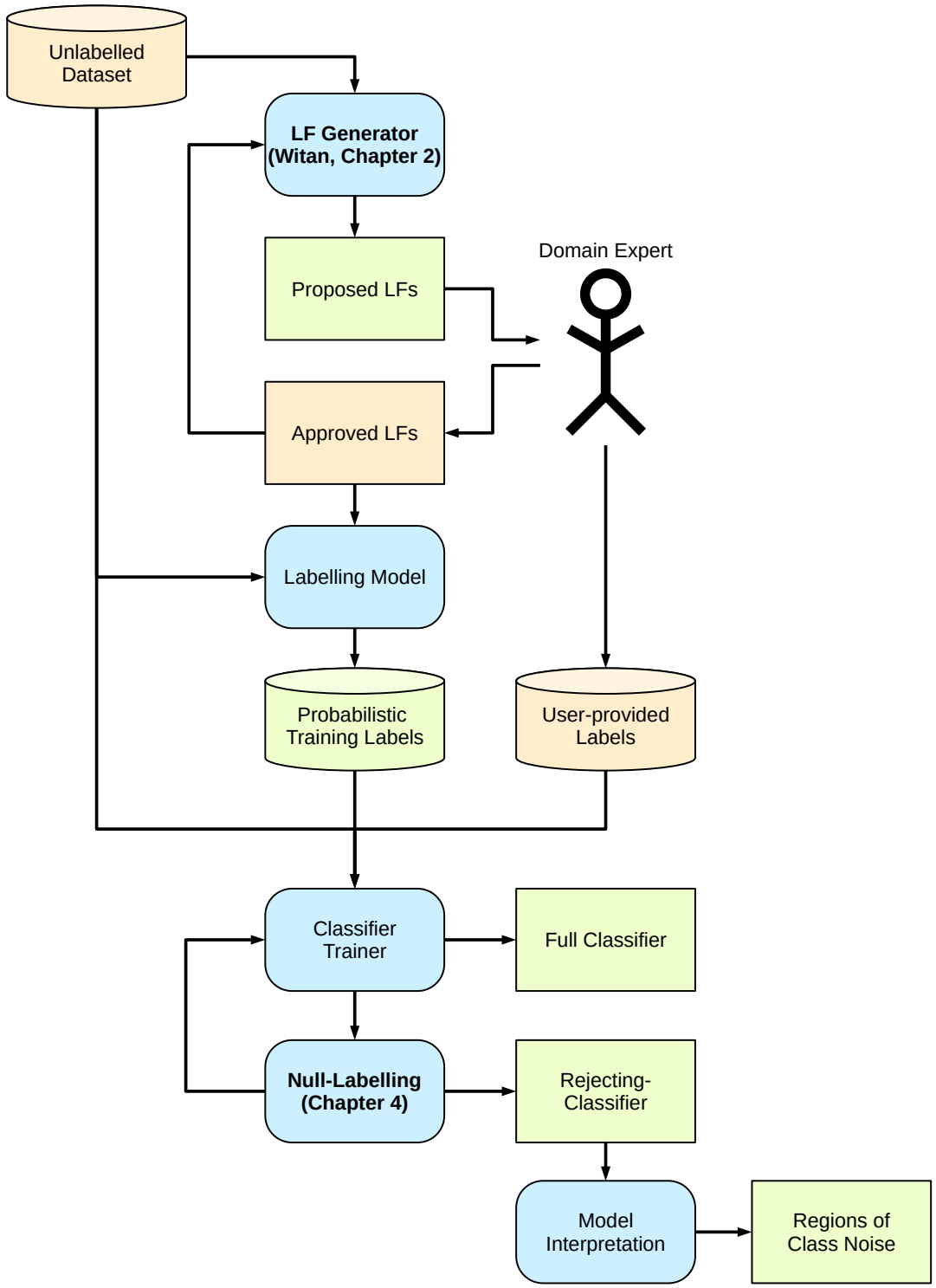


Figure 5.3: Logical view of components involved in classifier training.

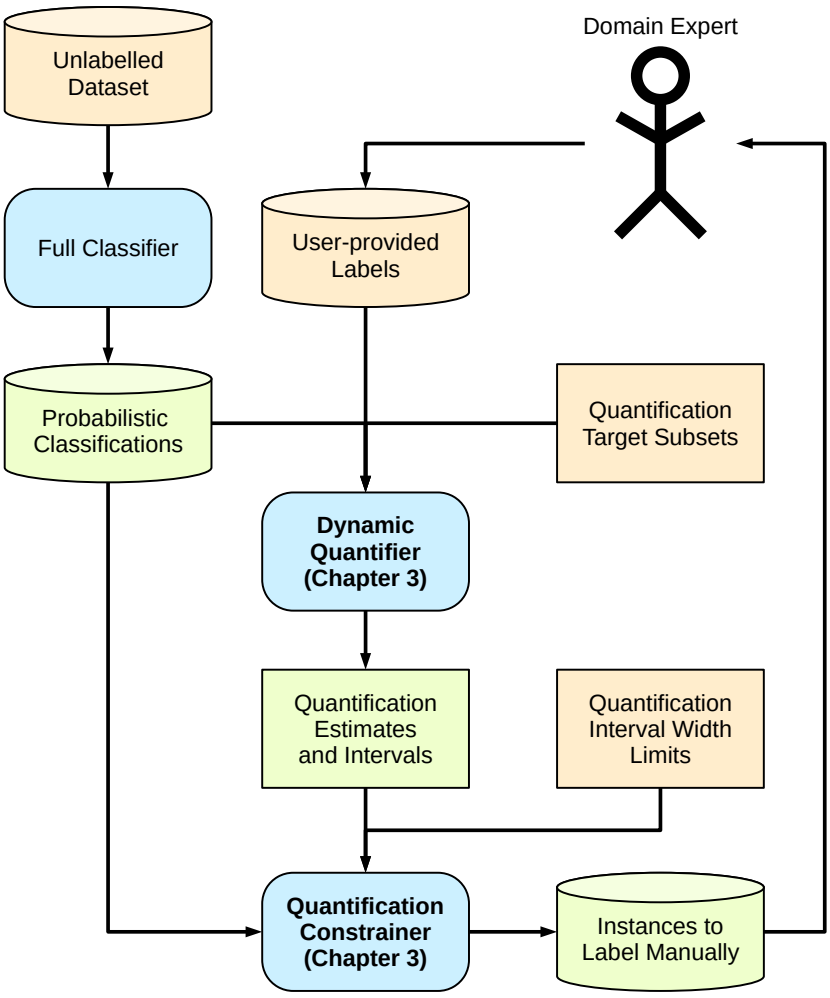


Figure 5.4: Logical view of components involved in classification and quantification.

5.2.1 Classifier Training

To support classifier training when the user lacks class labels for training data, the architecture applies the data programming paradigm as supported by the WITAN labelling function (LF) generator presented in Chapter 2. The ***LF Generator*** receives the full ***Unlabelled Dataset*** of instances the data consumer wishes to be classified or quantified and uses it to generate ***Proposed LFs***. For example, the ***Unlabelled Dataset*** could be a collection of unlabelled warranty claim descriptions that the data consumer wishes to categorise by fault type. The ***Domain Expert*** reviews ***Proposed LFs*** to produce a set of ***Accepted LFs*** by selecting LFs they believe to be more accurate than a random classifier, assigning class labels to those LFs. They may optionally tweak LF conditions or manually craft additional LFs. The ***LF Generator*** can optionally receive the current set of ***Accepted LFs*** to guide subsequent LF generation. Once the ***Domain Expert*** is content that the set of ***Accepted LFs*** assigns labels to a representative portion of the ***Unlabelled Dataset***, the ***Accepted LFs*** are applied to the ***Unlabelled Dataset*** within a ***Labelling Model*** (e.g. Snorkel; A. Ratner et al., 2017) that estimates the relative accuracies of LFs to produce a set of ***Probabilistic Training Labels***. A probabilistic class label is produced for each unlabelled data instance covered by an LF, comprising a numeric probability for each of m possible class labels: $(\tilde{y}_1, \ldots, \tilde{y}_m), \tilde{y}_i \in [0, 1]$. In the running example, probabilistic class labels would assign a probability for each possible fault type to each warranty claim covered by one or more LFs.

Once a set of ***Probabilistic Training Labels***, ***User-provided Labels***, or a combination of the two has been produced for a sufficient portion of the ***Unlabelled Dataset***, the instances and labels may be passed to the ***Classifier Trainer***. The ***Classifier Trainer*** is a standard supervised classification algorithm applied with a noise-aware loss function to account for the probabilistic labels. As many quantification algorithms expect classification outputs that approximate posterior probabilities $P(Y|X)$, the ***Classifier Trainer***

should either utilise a classification model that natively approximates probabilities (such as logistic regression) or include a post-calibration step (Esuli et al., 2020; Garg, Wu, Balakrishnan & Lipton, 2020). The ***Classifier Trainer*** will initially produce a ***Full Classifier*** that will assign a classification to all instances. In the running example, the classifier would be trained to assign a fault type to each warranty claim based on its textual description. Additionally, subsequent training iterations with Chapter 4's ***Null-Labelling*** applied to the training set can produce a ***Rejecting-Classifier*** that will reject to classify instances that cannot be accurately classified due to class noise, such as warranty claims whose descriptions do not adequately describe the underlying fault. Furthermore, the ***Region of Class Noise*** can be identified from the ***Rejecting-Classifier*** by performing ***Model Interpretation***, such as simple inspection of learned weights or more sophisticated methods (Molnar, 2022).

5.2.2 Classifier Application

Fig. 5.4 demonstrates the application of a trained classifier to produce both classifications and quantifications. Specifically, the non-rejecting ***Full Classifier*** is applied, as the process for constraining quantification intervals will perform its own rejection as necessary to meet user-specified ***Quantification Interval Width Limits***. The ***Full Classifier*** can provide ***Probabilistic Classifications*** for every instance in the ***Unlabelled Dataset***, generalising the patterns learned based on the training labels that may have only covered a subset of instances. The ***Dynamic Quantifier*** takes the full set of ***Probabilistic Classifications*** and produces per-class quantification estimates and intervals for each ***Quantification Target Subset*** required by the data consumer. In the running example, the quantifier would estimate the relative proportions of fault types within each target subset. The ***Dynamic Quantifier*** uses the statistical testing methodology from Chapter 3 to select an appropriate quantification method for each target subset.

Finally, the ***Quantification Constrainer*** applies the binary integer programming framework presented in Chapter 3 to ensure quantification intervals are within the ***Quantification Interval Width Limits*** required by the data consumer. It does so by identifying minimal sets of ***Instances to Label Manually***; once the ***Domain Expert*** has provided ***User-provided Labels*** for these instances, the ***Dynamic Quantifier*** can treat them as perfectly certain replacements of probabilistic classifications, thereby reducing quantification uncertainty and, consequently, interval widths. These ***User-provided Labels*** can also be used during classifier training to replace ***Probabilistic Training Labels*** produced by the ***Labelling Model***, though providing training labels for instances not previously labelled by labelling functions should be carefully considered, lest they bias the distribution of the training set.

5.2.3 Variability Guide

There are several points of variability that allow the architecture to be customised to suit the needs of specific applications. Firstly, The ***Labelling Model***, ***Classifier Trainer***, and quantification methods within the ***Dynamic Quantifier*** can all be independently modified or replaced to improve performance as the state-of-the-art evolves. Furthermore, some sections of the architecture can be omitted if the user does not require them. For example, the entire pipeline for generating ***Probabilistic Training Labels*** may be removed if the ***Domain Expert*** already has access to a sufficient set of ***User-provided Labels***.

5.3 Usage Scenarios

The flexibility of the proposed architecture is demonstrated in the following scenarios of how domain experts can interact with the system to provide inputs for training classifiers and how data consumers can apply the system to classify and quantify target data.

The WITAN LF generator supports a variety of interaction modes for producing sets of LFs. Firstly, WITAN can be used like other feedback-based assisted data programming methods (Giacometti & Soulet, 2017; Galhotra et al., 2021; Kartchner et al., 2020; Boecking et al., 2021) to generate one LF at a time for the domain expert to interactively review. Unlike other methods, WITAN can generate a set of LFs to be reviewed collectively, allowing the domain expert to have a clearer picture of the identified structure of the data. For example, the domain expert may find it easier to design a taxonomy of fault types for warranty claims after seeing a range of possible LFs that partition the data in various ways. After the user has reviewed a set of proposed LFs, WITAN can still be re-run to suggest additional LFs.

The domain expert can also supply pre-existing training artefacts to the system to assist classifier training. An initial set of *seed LFs* can be provided to guide WITAN towards generating complementary LFs. Any available ground-truth class labels can also be supplied to override the probabilistic training labels produced by the labelling model.

When applying the system, the data consumer can decide whether to apply the ***Full Classifier*** or ***Rejecting-Classifier*** for classification, depending on whether the application demands a classification for every instance and whether class noise is present and needs to be handled through rejection. Additionally, as new unlabelled data arrives to be classified and quantified over time, any instances matched by one or more LFs can be used to re-train the classifiers, allowing them to improve over time without additional effort from the user. Furthermore, if new data causes the distribution to shift, it will be detected by the dynamic quantifier and communicated to the data consumer through quantification intervals with increased widths. If the dynamic quantifier detects a major non-prior shift, the user may also wish to re-run WITAN to generate new potential labelling functions to cover any new sub-populations within the target data. Such a major shift may indicate the appearance of a previously unseen kind of fault or

a new cause of previously observed faults within a dataset of warranty claims, which may necessitate updates to the set of labelling functions under the guidance of a domain expert.

5.4 Conclusion

This chapter has presented an architecture for a tolerant machine learning system to perform supervised classification and quantification. The architecture is designed to produce results that meet user expectations of reliability despite training data deficiencies and to support a range of interaction modes for different forms of available data and knowledge inputs for training. The modular design also allows for the replacement of individual components as the state-of-the-art in machine learning improves. A system developed based on this architecture should allow users to begin deriving value from their data with minimal requirements of data quality and user effort.

Chapter 6

Conclusions and Future Directions

This thesis has contributed a number of novel methods for *tolerant machine learning*, enabling users to deploy and derive value from machine learning despite deficiencies in their training data. Specifically, contributions have been made to address the following training data deficiencies:

1. A lack of class labels for training data, including the absence of a set of possible classes (RO1). The proposed WITAN algorithm for assisted data programming without initial supervision addressed this deficiency (Chapter 2).
2. Dataset shift between the distributions of training data and target populations (RO2), particularly when quantifying the relative prevalences of different classes. This deficiency was addressed by a novel *Gain-Some-Lose-Some* (GSLS) quantification method for general dataset shift, a decision tree for dynamically selecting an appropriate quantification method given source and target data samples, and a framework for constraining quantification prediction intervals (Chapter 3).
3. Class noise that disrupts the relationship between input features and class labels or true class values (RO3). This deficiency was addressed by a novel, model-agnostic *null-labelling* method for classification with rejection (Chapter 4).

The experimentally demonstrated success of these contributions in addressing their target data deficiencies has demonstrated the viability of tolerant machine learning in such challenging circumstances. These contributions have also been integrated into a software architecture for a machine learning system that is tolerant of the full range of deficiencies and can therefore be deployed and relied upon with minimal requirements of input data quality and user effort (Chapter 5).

Tolerant machine learning is seen as an important step towards making machine learning a viable solution for potential users facing crippling data quality issues. It is hoped that this thesis will lead to continued work towards tolerant machine learning solutions that address even more common training data deficiencies, further increasing the potential for machine learning to deliver value in more diverse applications.

6.1 Limitations

There are some limitations to this research that could be addressed in future research.

Firstly, WITAN has only been evaluated through the use of a simulated user for reviewing labelling functions. While past research has indicated that users can effectively review labelling functions (Boecking et al., 2021), additional user studies could seek to understand to what degree the interpretability of labelling functions is impacted by WITAN's use of conjunctions and disjunctions in conditions.

Secondly, the primary evaluations of the proposed quantification and null-labelling rejections methods were performed under simulated conditions of dataset shift and class noise. This was due to the limited availability of real-world datasets with well-understood conditions of shift and class noise. Future research is needed to identify and categorise benchmark datasets that can be used to evaluate methods that address such data issues.

6.2 Directions for Future Work

The contributions made in this thesis have also created opportunities for future work, which can be categorised into the three broad research directions presented below.

6.2.1 More Diverse Applications of Tolerant Machine Learning

Firstly, there are opportunities to extend the developed tolerant machine learning methods to more diverse applications.

While experiments have been performed with WITAN using bag-of-words labelling functions for text classification tasks, WITAN could also be applied to other kinds of data, such as with labelling functions based on characteristics of images (as used by Boecking et al. (2021)) or numeric features. The challenge lies in developing binary labelling function conditions from richer data that can differentiate classes while being easily understood by domain experts. For example, labelling function conditions have previously been based on the results of clustering algorithms (Luo & Hauskrecht, 2018, 2019; Alonso Doval, 2021), but the inclusion criteria for clusters produced by common clustering methods are often not easy to understood. Consider centroid-based clustering: Except for datasets with very few dimensions, it is non-trivial to present centroid-based clusters to a user in a way that precisely communicates each cluster's boundaries.

Furthermore, null-labelling could be applied in the context of active learning to identify instances to query. Confidence scores are often used by active learning methods in the selection of instances to query (Yang & Loog, 2018), much like the selection of instances to reject with confidence-thresholding. An approach based on null-labelling could address the problems faced by confidence scores in the context of not-at-random class noise presented in this work. Knowledge gained by modelling the `null` class could help identify regions of the input space to be targeted by instance queries (or to be avoided if the noise is irreducible).

6.2.2 Improved Accuracy of Tolerant Machine Learning Methods

Secondly, the proposed tolerant machine learning methods could be improved or applied in new ways to improve model accuracy.

Given the hierarchical nature of WITAN's conjunctive labelling functions, classification accuracy may be improved by accounting for the inherent dependencies among functions in the labelling model. Furthermore, based on past successes combining weak supervision with active learning (Biegel, El-Khatib, Oliveira, Baak & Aben, 2021; Nashaat et al., 2018) and on the positive results with instance-level labelling for smaller classes in Section 2.5, it could be beneficial for multi-class and imbalanced tasks to combine WITAN with active-learning. This would allow a small set of labelled instances to act as specific "exceptions" to coarser labelling functions.

The knowledge gained by applying null-labelling to model class noise with an explicit `null` class could be leveraged to further improve models in a similar fashion to the probabilistic and model-based methods for addressing label noise (Frénay & Verleysen, 2013). For example, consider a dataset containing not-at-random label noise (the form of class noise where labels do not always accurately reflect the true class value) where multiple samples are available with different noise distributions, such as samples labelled by different experts or captured at different points in time under different conditions. A comparison of class noise models learned on different samples could be used to identify regions containing noisy instances in only some of the samples, such that the noisy instances could be removed to leave only trustworthy instances in those regions from other samples.

6.2.3 Tolerant Machine Learning for Broader Deficiencies

Finally, the proposed tolerant machine learning methods could be extended to account for a broader range of training data deficiencies.

Given WITAN's reliance on identifying statistical patterns in an unlabelled dataset, applying significance tests similar to those performed by the CN2 rule learner (Clark & Niblett, 1989) may improve labelling function suggestions for smaller unlabelled training datasets.

Future work on quantification under dataset shift could extend the dynamic quantification decision tree and framework for constrained intervals with additional quantification methods that provide specialised handling for additional subtypes of general dataset shift, such as covariate shift (Takahashi & Braga, 2020). Additionally, future work could seek to improve the fitting of GSLS under conditions of shift that were identified as particularly challenging. For example, an extended GSLS fitting method could attempt to account for unobservable but expected overlap between gain and loss distributions.

References

Agrawal, A., Verschueren, R., Diamond, S. & Boyd, S. (2018). A rewriting system for convex optimization problems. *Journal of Control and Decision*, *5*(1), 42–60.

Ahmed, H., Traore, I. & Saad, S. (2018). Detecting opinion spams and fake news using text classification. *Security and Privacy*, *1*(1), e9.

Alaíz-Rodríguez, R., Guerrero-Curieses, A. & Cid-Sueiro, J. (2011). Class and subclass probability re-estimation to adapt a classifier in the presence of concept drift. *Neurocomputing*, *74*(16), 2614–2623.

Alexandari, A., Kundaje, A. & Shrikumar, A. (2020). Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation. In *Proceedings of the 37th international conference on machine learning* (Vol. 119, pp. 222–232).

Almeida, T. A., Hidalgo, J. M. G. & Yamakami, A. (2011). Contributions to the study of sms spam filtering: new collection and results. In *Proceedings of the 11th acm symposium on document engineering* (pp. 259–262).

Alonso Doval, P. (2021). *Strategies for the programmatic generation of labelled corpus for text classification* (Unpublished master's thesis). University of Vigo.

Alrashdi, R. & O'Keefe, S. (2020). Automatic labeling of tweets for crisis response using distant supervision. In *Companion proceedings of the web conference 2020* (pp. 418–425). New York, NY, USA: Association for Computing Machinery.

Arachie, C. & Huang, B. (2021). Constrained labeling for weakly supervised learning. In *Uncertainty in artificial intelligence* (pp. 236–246).

Bae, J., Helldin, T., Riveiro, M., Nowaczyk, S., Bouguelia, M.-R. & Falkman, G. (2020). Interactive clustering: a comprehensive review. *ACM Computing Surveys (CSUR)*, *53*(1), 1–39.

Bella, A., Ferri, C., Hernández-Orallo, J. & Ramirez-Quintana, M. J. (2010). Quantification via probability estimators. In *2010 ieee international conference on data mining* (pp. 737–742).

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F. & Vaughan, J. W. (2010). A theory of learning from different domains. *Machine learning*, *79*(1), 151–175.

Bergner, B. & Krempl, G. (2016). Active subtopic detection in multitopic data. In *Proceedings of the workshop active learning: Applications, foundations and emerging trends, al@iknow 2016* (pp. 35–44).

Betham, W. (1834). The origin and history of the constitution of england: And of the early parliaments of ireland. In (p. 44). William Curry, Jun. And Co.

Biegel, S., El-Khatib, R., Oliveira, L. O. V. B., Baak, M. & Aben, N. (2021). Active weasul: Improving weak supervision with active learning. *arXiv preprint arXiv:2104.14847*.

Boecking, B., Neiswanger, W., Xing, E. & Dubrawski, A. (2021). Interactive weak supervision: Learning useful heuristics for data labeling. In *International conference on learning representations.*

Bootkrajang, J. & Chaijaruwanich, J. (2018). Towards instance-dependent label noise-tolerant classification: a probabilistic approach. *Pattern Analysis and Applications*, 1–17.

Bouvier, V., Very, P., Hudelot, C. & Chastagnol, C. (2019). Hidden covariate shift: A minimal assumption for domain adaptation. *arXiv preprint arXiv:1907.12299*.

Boyd, S., Boyd, S. P. & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Brodley, C. E. & Friedl, M. A. (1999). Identifying mislabeled training data. *Journal of artificial intelligence research*, *11*, 131–167.

Cachay, S. R., Boecking, B. & Dubrawski, A. (2021). Dependency structure misspecification in multi-source weak supervision models. *arXiv preprint arXiv:2106.10302*.

Chow, C. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, *16*(1), 41–46.

Cieslak, D. A. & Chawla, N. V. (2009). A framework for monitoring classifiers' performance: when and why failure occurs? *Knowledge and Information Systems*, *18*(1), 83–108.

Clark, P. & Niblett, T. (1989). The cn2 induction algorithm. *Machine Learning*, *3*(4), 261–283.

Cohen-Wang, B., Mussmann, S., Ratner, A. & Ré, C. (2019). Interactive programmatic labeling for weak supervision. In *Proceedings of the kdd data collection, curation, and labeling for mining and learning workshop.*

Cortes, C., DeSalvo, G. & Mohri, M. (2016). Boosting with abstention. In *Advances in neural information processing systems* (pp. 1660–1668).

Crowdflower. (2019). *Twitter us airline sentiment* (No. Version 4). Retrieved from `https://www.kaggle.com/crowdflower/twitter-airline-sentiment`

Das, S., Lade, P. & Srinivasan, S. (2020). Model adaptation and unsupervised learning with non-stationary batch data under smooth concept drift. *arXiv preprint arXiv:2002.04094*.

Daughton, A. R. & Paul, M. J. (2019). Constructing accurate confidence intervals when aggregating social media data for public health monitoring. In *International workshop on health intelligence* (pp. 9–17).

De-Arteaga, M., Romanov, A., Wallach, H., Chayes, J., Borgs, C., Chouldechova, A., ... Kalai, A. T. (2019). Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *Proceedings of the conference on fairness, accountability, and transparency* (pp. 120–128).

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, *7*, 1–30.

Denham, B., Lai, E. M., Sinha, R. & Naeem, M. A. (2021). Gain-some-lose-some: Reliable quantification under general dataset shift. In *2021 ieee international conference on data mining (icdm)* (pp. 1048–1053).

Denham, B., Lai, E. M., Sinha, R. & Naeem, M. A. (2022a). *Dynamic quantification with constrained error under unknown general dataset shift.* [Manuscript submitted for publication]. School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology.

Denham, B., Lai, E. M., Sinha, R. & Naeem, M. A. (2022b). Witan: unsupervised labelling function generation for assisted data programming. *Proceedings of the VLDB Endowment*, *15*(11), 2334–2347.

Denham, B., Pears, R. & Naeem, M. A. (2020). Null-labelling: A generic approach for learning in the presence of class noise. In *2020 ieee international conference on data mining (icdm)* (pp. 990–995).

dos Reis, D., Maletzke, A., Cherman, E. & Batista, G. (2018). One-class quantification. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 273–289).

dos Reis, D. M. (2020). *Non-stationary and unpredictable data distributions in classification and quantification* (Unpublished doctoral dissertation). Universidade de São Paulo.

dos Reis, D. M., Maletzke, A. G. & Batista, G. E. (2018). Unsupervised context switch for classification tasks on data streams with recurrent concepts. In *Proceedings of the 33rd annual acm symposium on applied computing* (pp. 518–524).

Du, J. & Ling, C. X. (2010). Asking generalized queries to domain experts to improve learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(6), 812–825.

Dua, D. & Graff, C. (2017). *UCI machine learning repository.* Retrieved from `http://archive.ics.uci.edu/ml`

Esuli, A., Molinari, A. & Sebastiani, F. (2020). A critical reassessment of the saerens-latinne-decaestecker algorithm for posterior probability adjustment. *ACM Transactions on Information Systems (TOIS)*, *39*(2), 1–34.

Evensen, S., Ge, C., Choi, D. & Çağatay Demiralp. (2020). Data programming by demonstration: A framework for interactively learning labeling functions. *arXiv preprint arXiv:2009.01444*.

Ferri, C. & Hernández-Orallo, J. (2004). Cautious classifiers. *ROCAI*, *4*, 27–36.

Forman, G. (2008). Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery*, *17*(2), 164–206.

Forman, G., Kirshenbaum, E. & Suermondt, J. (2006). Pragmatic text mining: minimizing human effort to quantify many issues in call logs. In *Proceedings of the 12th acm sigkdd international conference on knowledge discovery and data mining* (pp. 852–861).

Forman, G., Nachlieli, H. & Keshet, R. (2015). Clustering by intent: a semi-supervised method to discover relevant clusters incrementally. In *Joint european conference*

on machine learning and knowledge discovery in databases (pp. 20–36).

Frénay, B. & Verleysen, M. (2013). Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, *25*(5), 845–869.

Fu, D., Chen, M., Sala, F., Hooper, S., Fatahalian, K. & Ré, C. (2020). Fast and three-rious: Speeding up weak supervision with triplet methods. In *International conference on machine learning* (pp. 3280–3291).

Fumera, G. & Roli, F. (2004). Analysis of error-reject trade-off in linearly combined multiple classifiers. *Pattern Recognition*, *37*(6), 1245–1265.

Fumera, G., Roli, F. & Giacinto, G. (2000). Reject option with multiple thresholds. *Pattern recognition*, *33*(12), 2099–2101.

Fürnkranz, J. & Flach, P. A. (2005). Roc 'n'rule learning—towards a better understanding of covering algorithms. *Machine Learning*, *58*(1), 39–77.

Galhotra, S., Golshan, B. & Tan, W.-C. (2021). Adaptive rule discovery for labeling text data. In *Proceedings of the 2021 international conference on management of data* (pp. 2217–2225).

García-Zattera, M. J., Mutsvari, T., Jara, A., Declerck, D. & Lesaffre, E. (2010). Correcting for misclassification for a monotone disease process with an application in dental research. *Statistics in medicine*, *29*(30), 3103–3117.

Garg, S., Wu, Y., Balakrishnan, S. & Lipton, Z. (2020). A unified view of label shift estimation. *Advances in Neural Information Processing Systems*, *33*, 3290–3300.

Geifman, Y. & El-Yaniv, R. (2017). Selective classification for deep neural networks. In *Advances in neural information processing systems 30* (pp. 4878–4887). Curran Associates, Inc.

Giacometti, A. & Soulet, A. (2017). Interactive pattern sampling for characterizing unlabeled data. In *International symposium on intelligent data analysis* (pp. 99–111).

González, P., Castaño, A., Chawla, N. V. & Coz, J. J. D. (2017). A review on quantification learning. *ACM Computing Surveys (CSUR)*, *50*(5), 1–40.

González, P., Castaño, A., Peacock, E. E., Díez, J., Del Coz, J. J. & Sosik, H. M. (2019). Automatic plankton quantification using deep features. *Journal of Plankton Research*, *41*(4), 449–463.

González-Castro, V., Alaiz-Rodríguez, R. & Alegre, E. (2013). Class distribution estimation based on the hellinger distance. *Information Sciences*, *218*, 146–164.

Grechkin, M., Poon, H. & Howe, B. (2018). Ezlearn: Exploiting organic supervision in automated data annotation. In *Proceedings of the 27th international joint conference on artificial intelligence* (p. 4085-4091).

Gu, F., Zhang, G., Lu, J. & Lin, C.-T. (2016). Concept drift detection based on equal density estimation. In *2016 international joint conference on neural networks (ijcnn)* (pp. 24–30).

Hancock, B., Varma, P., Wang, S., Bringmann, M., Liang, P. & Ré, C. (2018). Training classifiers with natural language explanations. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 1884–1895).

Hanczar, B. (2019). Performance visualization spaces for classification with rejection option. *Pattern Recognition*, *96*, 106984.

Hassan, W., Maletzke, A. & Batista, G. (2020). Accurately quantifying a billion instances per second. In *2020 ieee 7th international conference on data science and advanced analytics (dsaa)* (pp. 1–10).

He, R. & McAuley, J. (2016). Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th international conference on world wide web* (pp. 507–517).

Hofer, V. (2015). Adapting a classification rule to local and global shift when only unlabelled data are available. *European Journal of Operational Research*, *243*(1), 177–189.

Jain, S., White, M., Trosset, M. W. & Radivojac, P. (2016). Nonparametric semi-supervised learning of class proportions. *arXiv preprint arXiv:1601.01944*.

Jiang, H., Kim, B., Guan, M. & Gupta, M. (2018). To trust or not to trust a classifier. In *Advances in neural information processing systems 31* (pp. 5541–5552). Curran Associates, Inc.

Johnson, B. A. & Iizuka, K. (2016). Integrating openstreetmap crowdsourced data and landsat time-series imagery for rapid land use/land cover (lulc) mapping: Case study of the laguna de bay area of the philippines. *Applied Geography*, *67*, 140–149.

Karamanolakis, G., Hsu, D. & Gravano, L. (2019). Leveraging just a few keywords for fine-grained aspect detection through weakly supervised co-training. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)* (pp. 4611–4621).

Kartchner, D., Ren, W., Nakajima An, D., Zhang, C. & Mitchell, C. S. (2020). Regal: Rule-generative active learning for model-in-the-loop weak supervision. In *Neurips 2020 workshop on human and model in the loop evaluation and training strategies.*

Keith, K. & O'Connor, B. (2018). Uncertainty-aware generative models for inferring document class prevalence. In *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 4575–4585).

Kruchten, P. B. (1995). The 4+1 view model of architecture. *IEEE software*, *12*(6), 42–50.

Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Machine learning proceedings 1995* (pp. 331–339). Elsevier.

Lemberger, P. & Panico, I. (2020). A primer on domain adaptation. *arXiv preprint arXiv:2001.09994*.

Lewis, D. D. & Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international acm-sigir conference on research and development in information retrieval* (pp. 3–12).

Lipton, Z. C., Wang, Y.-X. & Smola, A. (2018). Detecting and correcting for label shift with black box predictors. In *International conference on machine learning* (pp. 3122–3130).

Loukides, M. (2022). *Ai adoption in the enterprise.* O'Reilly Media. Sebastopol, CA: O'Reilly Media. Retrieved from https://www.oreilly.com/radar/ai-adoption-in-the-enterprise-2022/

Lu, B., Ott, M., Cardie, C. & Tsou, B. K. (2011). Multi-aspect sentiment analysis with topic models. In *2011 ieee 11th international conference on data mining workshops* (pp. 81–88).

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J. & Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, *31*(12), 2346–2363.

Luengo, J., Shim, S.-O., Alshomrani, S., Altalhi, A. & Herrera, F. (2018). Cnc-nos: Class noise cleaning by ensemble filtering and noise scoring. *Knowledge-Based Systems*, *140*, 27–49.

Luo, Z. & Hauskrecht, M. (2018). Hierarchical active learning with group proportion feedback. In *Proceedings of the 27th international joint conference on artificial intelligence* (pp. 2532–2538).

Luo, Z. & Hauskrecht, M. (2019). Hierarchical active learning with proportion feedback on regions. In *Machine learning and knowledge discovery in databases. ecml pkdd 2018. lecture notes in computer science* (Vol. 11052, pp. 464–480).

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y. & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies* (pp. 142–150).

Maletzke, A., Hassan, W., Reis, D. d. & Batista, G. (2020). The importance of the test set size in quantification assessment. In *Proceedings of the twenty-ninth international joint conference on artificial intelligence, IJCAI-20* (pp. 2640–2646).

Maletzke, A., Reis, D., Cherman, E. & Batista, G. (2018). On the need of class ratio insensitive drift tests for data streams. In *Second international workshop on learning with imbalanced domains: Theory and applications* (pp. 110–124).

Maletzke, A. G., dos Reis, D. M. & Batista, G. E. (2018). Combining instance selection and self-training to improve data stream quantification. *Journal of the Brazilian Computer Society*, *24*(1), 1–17.

Mallinar, N., Shah, A., Ho, T. K., Ugrani, R. & Gupta, A. (2020). Iterative data programming for expanding text classification corpora. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 34, pp. 13332–13337).

Mallinar, N., Shah, A., Ugrani, R., Gupta, A., Gurusankar, M., Ho, T. K., ... others (2019). Bootstrapping conversational agents with weak supervision. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 9528–9533.

Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Meng, Y., Shen, J., Zhang, C. & Han, J. (2018). Weakly-supervised neural text classification. In *Proceedings of the 27th acm international conference on information and knowledge management* (pp. 983–992).

Møller, B., Weedon-Fekjær, H. & Haldorsen, T. (2005). Empirical evaluation of prediction intervals for cancer incidence. *BMC Medical Research Methodology*,

5(1), 1–9.

Molnar, C. (2022). *Interpretable machine learning* (2nd ed.). Retrieved from https://christophm.github.io/interpretable-ml-book

Moreira dos Reis, D., Maletzke, A., Silva, D. F. & Batista, G. E. (2018). Classifying and counting with recurrent contexts. In *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining* (pp. 1983–1992).

Moreo, A. & Sebastiani, F. (2022). Tweet sentiment quantification: An experimental re-evaluation. *PLoS One*, *17*(9), e0263449.

Mouzannar, H., Rizk, Y. & Awad, M. (2018). Damage identification in social media posts using multimodal deep learning. In *The 15th international conference on information systems for crisis response and management (iscram)* (pp. 529–543).

Nadeau, C. & Bengio, Y. (1999). Inference for the generalization error. *Advances in neural information processing systems*, *12*.

Nashaat, M., Ghosh, A., Miller, J. & Quader, S. (2020). Asterisk: Generating large training datasets with automatic active supervision. *ACM Transactions on Data Science*, *1*(2), 1–25.

Nashaat, M., Ghosh, A., Miller, J., Quader, S., Marston, C. & Puget, J.-F. (2018). Hybridization of active learning and data programming for labeling large industrial datasets. In *2018 ieee international conference on big data (big data)* (pp. 46–55).

Neubauer, T. R., Peres, S. M., Fantinato, M., Lu, X. & Reijers, H. A. (2021). Interactive clustering: a scoping review. *Artificial Intelligence Review*, *54*(4), 2765–2826.

Ng, A. (2021, 3). *Mlops: From model-centric to data-centric ai.* Retrieved from https://web.archive.org/web/20220618040251/https://www.deeplearning.ai/wp-content/uploads/2021/06/MLOps-From-Model-centric-to-Data-centric-AI.pdf

Ni, C., Charoenphakdee, N., Honda, J. & Sugiyama, M. (2019). On the calibration of multiclass classification with rejection. In *Advances in neural information processing systems 32* (pp. 2586–2596). Curran Associates, Inc.

Prins, J., McCormack, D., Michelson, D. & Horrell, K. (2002). Product and process comparisons. In *Nist/sematech e-handbook of statistical methods* (chap. 7). Gaithersburg, MD, USA: National Institute of Standards and Technology. Retrieved from https://itl.nist.gov/div898/handbook/prc/section2/prc211.htm doi: 10.18434/M32189

Qin, H., Ma, X., Herawan, T. & Zain, J. M. (2014). Mgr: An information theory based hierarchical divisive clustering algorithm for categorical data. *Knowledge-Based Systems*, *67*, 401–411.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.

Rabanser, S., Günnemann, S. & Lipton, Z. (2019). Failing loudly: An empirical study of methods for detecting dataset shift. In *Advances in neural information processing systems* (Vol. 32).

Rajchl, M., Lee, M. C. H., Schrans, F., Davidson, A., Passerat-Palmbach, J., Tarroni, G., ... Rueckert, D. (2016). Learning under distributed weak supervision. *arXiv*

preprint arXiv:1606.01100.

Rashidi, P. & Cook, D. J. (2011). Ask me better questions: active learning queries based on rule induction. In *Proceedings of the 17th acm sigkdd international conference on knowledge discovery and data mining* (pp. 904–912).

Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S. & Ré, C. (2017). Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, *11*(3), 269–282.

Ratner, A. J., De Sa, C. M., Wu, S., Selsam, D. & Ré, C. (2016). Data programming: Creating large training sets, quickly. *Advances in Neural Information Processing Systems*, *29*, 3567–3575.

Rückert, U. & Kramer, S. (2004). Towards tight bounds for rule learning. In *Proceedings of the twenty-first international conference on machine learning* (p. 90).

Saerens, M., Latinne, P. & Decaestecker, C. (2002). Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. *Neural computation*, *14*(1), 21–41.

Schumacher, T., Strohmaier, M. & Lemmerich, F. (2021). A comparative evaluation of quantification methods. *arXiv preprint arXiv:2103.03223*.

Shah, K. & Manwani, N. (2019). Sparse reject option classifier using successive linear programming. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 4870–4877).

Shrikumar, A., Alexandari, A. & Kundaje, A. (2018). A flexible and adaptive framework for abstention under class imbalance. *arXiv preprint arXiv:1802.07024*.

Sousa, R. & Cardoso, J. S. (2013). The data replication method for the classification with reject option. *AI Communications*, *26*(3), 281–302.

Stumpf, S., Rajaram, V., Li, L., Burnett, M., Dietterich, T., Sullivan, E., … Herlocker, J. (2007). Toward harnessing user feedback for machine learning. In *Proceedings of the 12th international conference on intelligent user interfaces* (pp. 82–91).

Takahashi, C. C. & Braga, A. P. (2020). A review of off-line mode dataset shifts. *IEEE Computational Intelligence Magazine*, *15*(3), 16–27.

Tasche, D. (2016). Does quantification without adjustments work? *arXiv preprint arXiv:1602.08780*.

Tasche, D. (2017). Fisher consistency for prior probability shift. *The Journal of Machine Learning Research*, *18*(1), 3338–3369.

Tasche, D. (2019). Confidence intervals for class prevalences under prior probability shift. *Machine Learning and Knowledge Extraction*, *1*(3), 805–831.

Thulasidasan, S., Bhattacharya, T., Bilmes, J., Chennupati, G. & Mohd-Yusof, J. (2019). Combating label noise in deep learning using abstention. In *International conference on machine learning* (pp. 6234–6243).

Varma, P., Iter, D., De Sa, C. & Ré, C. (2017). Flipper: A systematic approach to debugging training sets. In *Proceedings of the 2nd workshop on human-in-the-loop data analytics* (pp. 1–5).

Varma, P. & Ré, C. (2018). Snuba: Automating weak supervision to label training data. *Proceedings of the VLDB Endowment*, *12*(3), 223–236.

Varma, P., Sala, F., He, A., Ratner, A. & Ré, C. (2019). Learning dependency structures for weak supervision models. In *International conference on machine learning* (pp. 6418–6427).

Vaz, A. F., Izbicki, R. & Stern, R. B. (2019). Quantification under prior probability shift: the ratio estimator and its extensions. *J. Mach. Learn. Res.*, *20*, 79–1.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., … SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. doi: 10.1038/s41592-019-0686-2

Wang, Z., Mekala, D. & Shang, J. (2021). X-class: Text classification with extremely weak supervision. In *Proceedings of the 2021 conference of the north american chapter of the association for computational linguistics: Human language technologies* (pp. 3043–3053).

Wei, W., Liang, J., Guo, X., Song, P. & Sun, Y. (2019). Hierarchical division clustering framework for categorical data. *Neurocomputing*, *341*, 118–134.

Wiener, Y. & El-Yaniv, R. (2011). Agnostic selective classification. In *Advances in neural information processing systems* (pp. 1665–1673).

Williams, H. P. (2009). Integer programming. In *Logic and integer programming* (pp. 25–70). Springer.

Yang, Y. & Loog, M. (2018). A benchmark and comparison of active learning for logistic regression. *Pattern Recognition*, *83*, 401–415.

Yildirim, M. Y., Ozer, M. & Davulcu, H. (2019). Leveraging uncertainty in deep learning for selective classification. *arXiv preprint arXiv:1905.09509*.

Zhang, J., Hsieh, C.-Y., Yu, Y., Zhang, C. & Ratner, A. (2022). A survey on programmatic weak supervision. *arXiv preprint arXiv:2202.05433*.

Zhang, J., Yu, Y., Li, Y., Wang, Y., Yang, Y., Yang, M. & Ratner, A. (2021). WRENCH: A comprehensive benchmark for weak supervision. In *Thirty-fifth conference on neural information processing systems datasets and benchmarks track.* Retrieved from `https://openreview.net/forum?id=Q9SKS5k8io`

Zhang, X., Zhao, J. & LeCun, Y. (2015). Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*, *28*, 649–657.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J. & Schölkopf, B. (2004). Learning with local and global consistency. In *Advances in neural information processing systems* (pp. 321–328).

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., … He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, *109*(1), 43–76.

Žliobaite, I. (2010). Change with delayed labeling: When is it detectable? In *2010 ieee international conference on data mining workshops* (pp. 843–850).

Appendix A

Application for Embargo

The Application for Embargo Form (PGR16) to embargo this thesis for 12 months from its lodgement is included in the following pages.

FORM PGR16 APPLICATION FOR EMBARGO

PLEASE NOTE

- ***This form must be typed. Handwritten forms will not be accepted.***
- ***Double clicking on the check boxes enables you to change them from not-checked to checked.***
- ***The completed form, signed by the student and the primary supervisor, should be submitted to the appropriate Faculty Postgraduate Office when the thesis/exegesis is lodged for examination. If the application is approved by the Faculty Postgraduate Committee, the form will be signed by the Dean (or delegate) and sent to the Graduate Research School for insertion into the print copies deposited. For more information consult the Postgraduate Handbook.***

Student ID	16957443	**Name**	Benjamin Denham
Faculty	**Design & Creative Technologies**	**School/Dept**	School of Engineering, Computer and Mathematical Sciences
Programme	Doctor of Philosophy	**Date of submission for examination**	11/11/2022
Research Output	**Thesis** ☒ **Dissertation** ☐	**Exegesis** ☐	**Points Value** 360
Research Title	Tolerant Machine Learning for Deficient Training Data		

EMBARGO TIMEFRAME

An embargo is requested on the public availability of the print and digital copies of the above thesis/exegesis from when the thesis is lodged in the Library (maximum normally 36).

12 **months**

EMBARGO CATEGORIES

The thesis/dissertation/exegesis contains confidential or sensitive information which if publicly available may ***(Tick all that apply)***

☐ Jeopardise the future intellectual property rights of the author (e.g. a patent application or publication)

☒ Breach a prior contractual arrangement with an external organisation *(Please attach a copy of the relevant agreement(s))*

☐ Infringe or endanger the right to privacy or cultural respect of an individual or group

The embargo would apply to

☒ The complete thesis/dissertation/exegesis

☐ A portion of the work (specify) :

SIGNATURES

Student	Benjamin Denham	**Signature**	Benjamin Denham	**Date**	9/11/2022
Primary Supervisor	Edmund Lai	**Signature**		**Date**	9 Nov 2022
Mentor of Primary Supervisor		**Signature**		**Date**	

RESTRICTED ACCESS APPROVED BY FACULTY DEAN(or delegate)

Signature Rosser Johnson **Date** 17-11-22

LIBRARY USE **RELEASE DATE**