Binary quantification in non-stationary scenarios

André Gustavo Maletzke

Tese de Doutorado do Programa de Pós-Graduação em Ciências de
Computação e Matemática Computacional (PPG-CCMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

André Gustavo Maletzke

Binary quantification in non-stationary scenarios

Doctoral dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Doctorate Program in Computer Science and Computational Mathematics. *EXAMINATION BOARD PRESENTATION COPY*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Gustavo E.A.P.A. Batista

USP – São Carlos
November 2019

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

M247b Maletzke, André Gustavo
Binary quantification in non-stationary
scenarios / André Gustavo Maletzke; orientador
Gustavo Batista. -- São Carlos, 2019.
163 p.

Tese (Doutorado - Programa de Pós-Graduação em
Ciências de Computação e Matemática Computacional) --
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2019.

1. Counting. 2. Machine Learning. 3. Concept
Drift. 4. Mixture Models. 5. Mosquito Traps. I.
Batista, Gustavo, orient. II. Título.

André Gustavo Maletzke

Quantificação binária em cenários não estacionários

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *EXEMPLAR DE DEFESA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Gustavo E.A.P.A. Batista

USP – São Carlos
Novembro de 2019

*Este trabalho é dedicado aos meus pais,
Irno e Marli, e a minha amada
esposa Barbara.*

ACKNOWLEDGEMENTS

Ao professor, orientador e amigo, Gustavo Enrique de Almeida Prado Alves Batista, pessoa que tenho profunda admiração e respeito. Gustavo, obrigado por me receber no ICMC/USP, por ceder o seu tempo, compartilhar ideias, me colocar em contato com pessoas incríveis e abrir portas não somente pra mim, mas também para a Barbara. Obrigado por tornar essa jornada tão bela e enriquecedora. Embora continuaremos em contato, sentirei saudades desse período.

Diversas pessoas compartilharam desta jornada comigo, mas duas delas foram além, compartilharam tempo, ideias e trabalharam em conjunto intensamente. Denis dos Reis e Everton Cherman, vocês foram decisivos para o desenvolvimento deste trabalho, aprendi muito com vocês. Grandes mentes, curiosos e sempre muito críticos e, é claro, grandes amigos.

Agradeço à minha família por me apoiar e incentivar nas minhas escolhas. Em especial e com muito carinho aos meus pais, Irno e Marli. A minha esposa Barbara, por compartilhar sonhos e trilhar comigo esse caminho, pelo seu amor e paciência.

À Chris e Everton, por todos os momentos incríveis compartilhados e que foram muitos. Sentirei saudades desse tempo.

Aos amigos, Gisele e Bruce, Thainara e Vinícius, Diego e Camila, por todos os momentos compartilhados.

A Sarah e Fábio, pela amizade, incentivo e por sempre abrirem as portas da sua casa durante esses quatro anos. A Nancy e Elio, por sempre incentivar cada escolha e por sempre me receber com carinho.

A Maria Carolina Monard, pessoa incrível, inteligência e senso crítico ímpar. Obrigado por sempre me receber, pelas conversas e compartilhar seu tempo comigo.

À Claudia Milare pelo incentivo e contribuição para o desenvolvimento deste trabalho.

Aos colegas e amigos do Labic que foram tantos/as durante esses quatro anos.

Ao professor Juliano Corbi por colaborar nas coletas de dados de diferentes espécies de insetos.

À professora Solange Rezende por sua dedicação e carinho com sempre nos recebe e pelo sua incansável busca por criar um ambiente de crescimento pessoal e profissional sempre melhor.

Professor Michael Reiskind, who received me in his lab during my internship abroad. His knowledge of mosquitoes identification, trapping, and population estimation has enriched

our algorithm proposals. I also would like to thank my co-workers and friends in Raleigh: Drew, Anastasia, Paul, Shawn, Steve, Meredith, Charles, John, Gabriela, and Professor Wes.

A Eduardo Hatano, pela amizade e contribuição na coleta de dados em Raleigh.

Ao Edi Samuel B. Mendonça pelo cooperação em diversas atividades durante todo o período.

Ao Instituto de Ciências Matemáticas e de Computação (ICMC) pela infraestrutura e pelos Professores e Funcionários o compõem.

À Universidade Estadual do Oeste do Parana (Unioeste).

I would like to thank the United States Agency for International Development under Grant No AID-OAA-F-16- 00072.

At last, I would like to thank Google for supporting us through the Research Latin America Award.

RESUMO

MALETZKE, A. G. **Quantificação binária em cenários não estacionários**. 2019. 163 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

Quantificação é uma tarefa de Aprendizado de Máquina semelhante à classificação, na qual o aprendizado é realizado a partir de um conjunto de treinamento com dados rotulados. Entretanto, a quantificação não está interessada em prever a classe de cada observação, mas em medir a representatividade de cada classe no conjunto de teste. A diferença sutil entre classificação e quantificação requer algoritmos específicos, medidas de desempenho e projetos experimentais diferentes. Além disso, a maioria dos algoritmos de quantificação existentes foi desenvolvida para cenários bem controlados que se baseiam no pressuposto de que a única mudança entre dados de treinamento e teste está na probabilidade *a priori* das classes. Esta tese concentra-se em fornecer melhorias nos algoritmos de quantificação, bem como no design experimental, incluindo suposições mais realistas. Especificamente, as principais contribuições desta tese são: (i) o primeiro algoritmo para quantificar dados não estacionários sob a presença de mudança de conceito; (ii) um detector de mudança de conceito não-supervisionado e insensível ao desbalanceamento de classe; (iii) um *framework* baseado em modelos de mistura juntamente com um extensivo estudo experimental, redefinindo a parametrização desse tipo de método; (iv) estudo sobre como o tamanho do *batch* altera o *ranking* dos algoritmos de quantificação e, consequentemente, a proposta de um *framework* baseado em meta-aprendizado para selecionar dinamicamente o melhor quantificador; (v) uma análise descritiva do impacto da variabilidade na qualidade dos scores sobre quantificadores existentes e, como resposta, a essa influência foi proposto um novo algoritmo que quantifica com precisão sob cenários de mudanças na qualidade dos *scores*, e; (vi) a aplicabilidade das propostas em um problema do mundo real, bem como os esforços para desenvolver um novo mecanismo de captura e contagem populacional de mosquitos.

Palavras-chave: contagem, aprendizado de máquina, mudança de conceito, modelos de mistura, armadilhas de mosquito.

ABSTRACT

MALETZKE, A. G. **Binary quantification in non-stationary scenarios**. 2019. 163 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

Quantification is a Machine Learning task similar to classification in the sense that it learns from a training set with labeled data. However, quantification is not interested in predicting the class of each observation, but rather measure the representativeness of each class in the test set. This subtle difference between classification and quantification requires specific algorithms, performance measures, and experimental designs. Moreover, most of the existing quantification algorithms were developed for well-controlled scenarios that rely on the assumption that the only change from training to test data is in the prior probability of the classes. This thesis focuses on providing improvements in quantification algorithms as well as the experimental design, including more realistic assumptions. Specifically, the main contributions of this thesis are the following: (i) the first algorithm to quantify non-stationary data under the concept drift presence; (ii) an unsupervised drift detector that are insensible to class imbalance explicitly; (iii) a mixture model framework for quantification with a deep experimental study, redefining the best parametrization of this sort of method; (iv) we show that the batch size, an ignored question in the literature, changes the ranking of quantification algorithms and thus we proposed a metalearning framework to select the best quantifier dynamically; (v) we describe how existing quantifiers are affected under score quality variability and, as response of it, we proposed a new algorithm that accurately quantifies while allowing changes in the quality of the scores, and; (vi) we show the applicability of our proposals in real-world problem and the efforts to contribute with development of new mechanism for trapping mosquitoes.

Keywords: counting, machine learning, concept drift, mixture models, mosquito traps.

LIST OF FIGURES

Figure 1 – Difference between the classification and quantification.	24
Figure 2 – Optical sensor used to capture mosquitoes sound.	26
Figure 3 – Applicability of the optical sensor embedded in traps.	27
Figure 4 – The wing-beat frequency variability	28
Figure 5 – Schematic design of the SQSI-IS algorithm	46
Figure 6 – Mean quantification errors using RF and CC	50
Figure 7 – Mean quantification errors using RF and PCC	51
Figure 8 – Mean quantification errors using RF and ACC	52
Figure 9 – Positive class distribution histograms	53
Figure 10 – Mean quantification errors using SVM and CC	55
Figure 11 – Mean quantification errors using SVM and PCC	56
Figure 12 – Mean quantification errors using SVM and ACC	57
Figure 13 – Histograms representing the circadian rhythm	62
Figure 14 – A comparison between two Random Forest classifiers on insect data	63
Figure 15 – Schematic representation of the HDy algorithm	67
Figure 16 – Distributions of the HDy distances without concept drifts.	68
Figure 17 – Influence of temperature on the wing-beat frequency	70
Figure 18 – Distance distribution of examples from two different concepts	71
Figure 19 – Experimental setup used to evaluate the CDT proposal	72
Figure 20 – Impact of class distribution over the drift detection accuracy	74
Figure 21 – Impact of class distribution over the classification accuracy	74
Figure 22 – Mean absolute quantification error obtained with DyS and Topsøe	89
Figure 23 – Mean absolute quantification error obtained with DyS and ORD	90
Figure 24 – Frequencies of the number of bins that produced the smallest error	90
Figure 25 – Rank positions for different distances in DyS with tuned bins	91
Figure 26 – Rank positions for different distances in DyS for a range of bins	92
Figure 27 – Comparison between SORD and Topsøe for varying test sample size	93
Figure 28 – Comparison between ORD and Topsøe for varying test sample size	93
Figure 29 – Nemenyi’s post hoc test for mean absolute quantification error	97
Figure 30 – Threshold and score distribution for a binary classification process	101
Figure 31 – Increase in positive cases make CC underestimates this class	101
Figure 32 – Framework architecture for quantifiers recommendation.	109
Figure 33 – Nemenyi’s post hoc test for mean absolute quantification error	112

Figure 34 – Mean absolute error variability for test set size and class distribution variability.	113
Figure 35 – Nemenyi’s post hoc test for mean absolute quantification error	114
Figure 36 – Mean absolute quantification error obtained with DyS+PS and MS	115
Figure 37 – Rank positions for all quantification algorithms	116
Figure 38 – Critical difference diagram among all algorithms	116
Figure 39 – Rank positions for all quantification algorithms, including MLQ framework	117
Figure 40 – Mean absolute quantification error of MLQ, TOP, and RND	118
Figure 41 – DySyn schematic representation	127
Figure 42 – Mean absolute error when m_{Tr} and m_{Te} vary	131
Figure 43 – Rank positions for all quantification algorithms in all tested settings	133
Figure 44 – Mean absolute error for different ratios of the first subclass	133
Figure 45 – Rank positions for all quantification algorithms	134
Figure 46 – Cylindrical chamber with the optical sensor embedded.	136
Figure 47 – Influence of temperature on the wing-beat frequency	137
Figure 48 – Chamber used to control temperature and humidity.	138
Figure 49 – A signal gathered when an <i>Aedes aegypti</i> (female) crosses the line-of-sight .	139
Figure 50 – Distribution of each species	139
Figure 51 – Distribution of our data collection over both temperature and humidity . . .	140
Figure 52 – Density functions of different insect species	141
Figure 53 – Examples of insect traps	144
Figure 54 – Smart trap with the optical sensor embedded	145
Figure 55 – Cage used in the Smart trap trials	146
Figure 56 – A simple experiment with a classifier embedded into the cylindrical chambers	147
Figure 57 – Events recorded in the trap during each experiment	148

LIST OF ALGORITHMS

Algorithm 1 – Self-training	44
Algorithm 2 – SQSI with instance selection and semi-supervised learning	47
Algorithm 3 – Ordinal Distance	83
Algorithm 4 – SORD Dissimilarity Function	85
Algorithm 5 – Generating test samples	108

LIST OF TABLES

Table 1 – Mean quantification error for each instance selection techniques using RF . . .	54
Table 2 – Statistical analysis between the active learning techniques using RF	54
Table 3 – Mean quantification error for each instance selection techniques using SVM . . .	55
Table 4 – Statistical analysis between the active learning techniques using SVM	56
Table 5 – Quantification error and statistical analysis	58
Table 6 – Mean accuracy rates for concept identification and classification	73
Table 7 – Dissimilarity Functions.	82
Table 8 – Datasets description.	88
Table 9 – Smallest upper bound for the number of histogram bins	91
Table 10 – Quantifiers evaluated.	111
Table 11 – Datasets description.	112
Table 12 – Datasets description.	129
Table 13 – Mean absolute error for each dataset and quantifier	132
Table 14 – Standard deviation of values presented in Table 13.	132
Table 15 – Performance of the Smart trap	148
Table 16 – Absolute error of different quantifiers	149

CONTENTS

1	INTRODUCTION	23
1.1	Quantification relevance	25
1.2	Main contributions	29
1.3	Thesis organization	30
1.3.1	Chapter 2	30
1.3.2	Chapter 3	31
1.3.3	Chapter 4	32
1.3.4	Chapter 5	33
1.3.5	Chapter 6	34
1.3.6	Chapter 7	34
1.3.7	Chapter 8	34
2	COMBINING INSTANCE SELECTION AND SELF-TRAINING TO IMPROVE DATA STREAM QUANTIFICATION	35
2.1	Introduction	35
2.2	Background	38
2.2.1	Data stream	38
2.2.2	Quantification	39
2.2.2.1	<i>Classify and Count</i>	<i>39</i>
2.2.2.2	<i>Probabilistic Classify and Count</i>	<i>40</i>
2.2.2.3	<i>Adjusted Classify and Count</i>	<i>40</i>
2.2.3	Evaluation metrics	40
2.2.4	Instance selection and semi-supervised learning	41
2.2.4.1	<i>Random Sampling</i>	<i>42</i>
2.2.4.2	<i>Farthest-First Traversal</i>	<i>42</i>
2.2.4.3	<i>Cluster-based Sampling</i>	<i>43</i>
2.3	Reference approaches for data stream quantification	44
2.3.1	Static	44
2.3.2	Sliding	44
2.4	The original and extended SQSI algorithm	45
2.5	Time complexity analysis	46
2.6	Experiments	48
2.7	Results	50

2.8	Limitations	58
2.9	Conclusions	59
3	ON THE NEED OF CLASS RATIO INSENSITIVE DRIFT TESTS FOR DATA STREAMS	61
3.1	Introduction	61
3.2	Related work	64
3.2.1	<i>Data streams and concept drift detection</i>	<i>64</i>
3.2.2	<i>Quantification</i>	<i>66</i>
3.3	Proposal	67
3.4	Experimental evaluation	69
3.5	Results	73
3.6	Conclusion	75
4	DYS: A FRAMEWORK FOR MIXTURE MODELS IN QUANTIFICATION	77
4.1	Introduction	78
4.2	Related work	79
4.3	DyS	81
4.3.1	<i>Histogram dissimilarity</i>	<i>82</i>
4.3.2	<i>Mixable Kolmogorov Smirnov</i>	<i>84</i>
4.3.3	<i>SORD</i>	<i>84</i>
4.4	Experimental setup	85
4.4.1	<i>Datasets</i>	<i>87</i>
4.5	Experimental evaluation	88
4.6	Conclusions and future work	94
5	THE IMPORTANCE OF THE TEST SET SIZE IN QUANTIFICATION ASSESSMENT	95
5.1	Introduction	95
5.2	Background	98
5.3	Related work	99
5.3.1	<i>Quantification methods</i>	<i>100</i>
5.3.1.1	<i>Classify and Count</i>	<i>100</i>
5.3.1.2	<i>Adjust Classify Count</i>	<i>102</i>
5.3.1.3	<i>Probabilistic Classify and Count</i>	<i>102</i>
5.3.1.4	<i>Probabilistic Adjust Classify and Count</i>	<i>103</i>
5.3.1.5	<i>Threshold selection methods</i>	<i>103</i>
5.3.1.6	<i>Expectation Maximization Quantification</i>	<i>103</i>
5.3.1.7	<i>Quantification Trees</i>	<i>104</i>

5.3.1.8	<i>HDy algorithm</i>	105
5.3.1.9	<i>DyS framework</i>	106
5.3.2	<i>How test set sizes affect the performance of the quantification methods</i>	106
5.3.3	<i>Experimental design</i>	107
5.4	Designing experiments in quantification	108
5.5	Metalearning quantification	108
5.6	Experimental setup	110
5.7	Experimental evaluation	112
5.8	Conclusion	117
6	ACCURATELY QUANTIFYING UNDER SCORE VARIABILITY	119
6.1	Introduction	119
6.2	Background	121
6.3	Brief survey of quantification methods	122
6.4	Proposed methods	124
6.4.1	<i>Model for Score Simulation</i>	125
6.4.2	<i>Distribution y-Similarity using Synthetic Scores</i>	126
6.5	Experimental setup	127
6.6	Experimental evaluation	130
6.7	Conclusion	133
7	OTHER CONTRIBUTIONS	135
7.1	Flying insect dataset	136
7.2	Smart trap	143
7.2.1	<i>Experiments</i>	143
7.2.2	<i>Results</i>	147
7.3	Collaborations in other areas	149
8	CONCLUSION	151
	BIBLIOGRAPHY	155

INTRODUCTION

Machine learning methods have seen a remarkable advance in their theoretical basis and practical utilization, with notable contributions to several tasks such as classification, clustering, regression and pattern description or recognition. Those tasks have enabled plenty of real-world applications from email spam and malware detection, facial and voice recognition, stock prices prediction, to diseases diagnosis.

Most of supervised machine learning applications consist in classifying or making predictions about individual events. Nevertheless, there are several applications where the objective is to estimate the statistics of groups, *i.e.*, problems where individual predictions are less relevant than the overall view. For instance, to answer questions such as: *how many positive and negative reviews a specific product received in the last month?* To provide global estimates such as the class distribution of a data sample is the role of a recently formalized learning task named *quantification*. Figure 1 illustrates the basic difference between quantification and classification. While the classifier pursues the correct estimation of the actual label of each instance in the test set, the quantifier is interested in estimating the class distribution of the test set.

Quantification is a prominent task in machine learning that has received an increasing amount of attention in the last years. The objective is to predict the class distribution in a data sample. Therefore, quantification finds applications in domains such as sentiment analysis, where the objective is to estimate the proportion of positive tweets about a particular subject (GONZÁLEZ *et al.*, 2017b). Hopkins and King (2010) used quantification to estimate the support for political candidates from blog posts. Quantification was also used to count different issues from call logs, aiming to optimize human resources according to the most required issues (FORMAN, 2008). Another example is mosquito surveillance, where the goal consists in calculating the number of mosquito species captured by an insect trap (SILVA *et al.*, 2015a).

Quantification and classification are also related tasks that benefit from each other.

Figure 1 – Difference between the classification and quantification.

Source: Elaborated by the author.

Quantification can be straightforwardly achieved by counting the number of examples that are classified as each class. This approach is known as Classify and Count (CC). CC has critical shortcomings, including a systemic error introduced when the class distribution drifts (DIETTERICH; KONG, 1995). A more reliable approach is Adjusted Classify and Count (ACC) (FORMAN, 2005), that fixes the bias in CC by using a correction factor based on the classifier's true (*tpr*) and false (*fpr*) positive rates. In practice, several factors, such as lack of data and class imbalance, hinder the estimation of these rates, impacting ACC as well. To mitigate that, Forman (2006) has proposed several strategies based on threshold selection policies that compose the state-of-the-art in binary quantification. On the other hand, classification benefits from quantification as an estimate of the class distribution is required to perform calibration (VUK; CURK, 2006).

Since the introduction of CC and ACC, the quantification community has proposed a large number of quantification methods. A family of algorithms that stands out due to its quantification accuracy, simplicity, and ability to work on imbalanced training samples is known as Mixture Models (MM). Initially proposed by Forman (2005), it also later extended González-Castro, Alaiz-Rodríguez and Alegre (2013), who adopted Hellinger distance and histogram of scores as representation of the data, in the HDy algorithm. Shortly, MM quantification comprises approaches where the probability distribution of each class is modeled individually and learned from a training set. The idea is to mix the distributions of positive and negative training data, at different rates, in order to find the mixture that is the least dissimilar to the distribution of the test data.

Although many improvements in quantification have been proposed over the years, the task remains relatively unexplored in real-world problems even when machine learning experts are involved. Consequently, quantification learning is also rarely included in traditional machine learning literature. These facts have produced some consequences: (a) most of quantification problems are tackled with the simplest and ineffective algorithm that consists in classifying and counting; (b) quantification algorithms are commonly designed and evaluated over well-controlled scenarios that underestimate real-world problems (c); the absence of a robust experimental design to evaluate quantification algorithms; and (d) different nomenclatures are used to describe quantification learning.

This thesis summarizes research results involving binary quantification algorithms. In this sense, this work focused on developing algorithms, experimental design formalization, and algorithms for non-stationary datasets. The main contributions of this thesis are: (i) the first algorithm to quantify non-stationary data under the concept drift presence; (ii) an unsupervised drift detector that is insensible to class imbalance explicitly; (iii) a mixture model framework for quantification with a deep experimental study, redefining the best parametrization of this sort of method; (iv) we show that the batch size, an ignored question in literature, changes the ranking of quantification algorithms and thus we proposed a metalearning framework to select the best quantifier dynamically; (v) we describe how existing quantifiers are affected under score quality variability and, as a response, we propose a new algorithm that accurately quantifies while allowing changes in the quality of the scores, and; (vi) we show the applicability of our proposals in real-world problem and the efforts to contribute with development of new mechanism for trapping mosquitoes. All contributions are presented in the next chapters but, preceded only by the necessary background and related works are presented.

The remaining of this chapter is organized as follows. Section 1.1 introduces necessary assumptions to define a quantification problem. Sections 1.2 and 1.3 describe the main contributions of this thesis and the organization for the remaining chapters, respectively.

1.1 Quantification relevance

Several real-world problems can be framed as quantification. For instance, estimating the percentage of positive and negatives reviews of a specific service or product across different periods of time; counting the number of positive, negative, or neutral tweets related to an interesting topic; and estimating how the traffic flow is distributed for distinct types of vehicles and many others applications.

Concrete applications in quantification have been applied in the last years. [González-Castro, Alaiz-Rodríguez and Alegre \(2013\)](#) evaluated quantification aiming to estimate the distribution of sperm cells, considering they can be intact or have damaged acrosome. Quantification algorithms were also used by [Beijbom et al. \(2015\)](#) on two large scale datasets from

marine ecology. Estimative of tweet distribution was explored in experiments conducted by [Gao and Sebastiani \(2016\)](#), and experiments over consumers' opinion were discussed by [González et al. \(2017b\)](#).

In this work, we are specifically interested in counting flying insects, especially mosquitoes. Our interest is motivated by the tremendous increase in the incidence of diseases transmitted by mosquitoes, such as dengue fever, chikungunya, and zika fevers. Mosquitoes are a global concern due to their high adaptability to different habitats and their rapid geographical expansion ([CARLSON; DOUGHERTY; GETZ, 2016](#)). Mosquito traps are the most known and used surveillance strategy, basically composed of a specific lure to attract and capture female mosquitoes. Most of the traps require a posterior analysis conducted by a specialist to identify and count the mosquitoes trapped, which is an expensive and time-consuming task. In the last decade, many efforts have been done to design an automatic mechanism to identify different species of mosquito using the sound they emit, motivated by results reported at the mid-20th century ([KAHN et al., 1945](#); [WISHART; RIORDAN, 1959](#)).

Oppositely, [Batista et al. \(2011\)](#) proposed a low-cost *optical* sensor to capture the sounds produce by flying insects. Figure 2 illustrates the design of the sensor.

Figure 2 – Optical sensor used to capture mosquitoes sound.

Source: Elaborated by Dr. Gustavo Batista.

This sensor is composed of a couple of parallel mirrors and an infrared LED that projects a curtain of light. When a flying insect crosses the light, this curtain is obstructed according to the size, number of wings, and movement of the wings. This entire light variation is captured by a phototransistor and converted into an audio signal in a process equivalent to the usage of a regular microphone, having the advantage of being deaf against any agent that does not cross the

light curtain.

The idea is to embed the optical sensor in traps and distributed them geographically, as illustrated in Figure 3. With this setting, we expect to estimate mosquito activity in real-time and consequently provide better data about mosquito populations. Crossing data from multiple traps can provide a mosquito density map, which can be turned into a layer in services such as Google Maps or Earth Engine, aiding mosquito control agencies to plan interventions.

Figure 3 – Applicability of the optical sensor embedded in traps.

Source: Elaborated by the author.

According to [Salim *et al.* \(2017\)](#), and [Villarreal, Winokur and Harrington \(2017\)](#), mosquito sounds are influenced by environmental conditions, mainly temperature, and physiologic aspects, primarily weight and wing size. Figure 4 illustrates the influence of temperature on the wing-beat frequency, one of the features we extract from the signals.

Mosquitoes can also use sounds like a communication signal (ARTHUR *et al.*, 2014), incurring greater variability of sound within a same species. Moreover, there are thousands species of mosquitoes and other flying insects, which makes any identification task a challenging effort.

For example, consider the female *Aedes aegypti* as our target in a identification task. In this case, the negative class (non-female *Aedes aegypti*) comprises all flying insects that can possibly pass in front of the sensor and are not female *Aedes aegypti*. Under this circumstance, we emphasize the fact of the number of insect species in nature is around ten million (CHAPMANN, 2004). Therefore, the negative class contains several known and unknown subclasses whose distributions vary independently.

Those factors make the estimate of mosquitoes population even more complex and impose scenarios that most of the quantification algorithms were not designed for. In general, quantification is a task similar to classification in the sense that we are provided with a training

Figure 4 – Influence of temperature on the wing-beat frequency of female *Aedes aegypti* mosquitoes.

Source: [Reis et al. \(2018\)](#).

set with labeled observations, and a test sample with unlabeled examples. Indeed, most of quantification algorithms rely on a classifier in preliminary steps and keep the assumptions required by the classifier.

For instance, classification methods assume that both training and test data are drawn from a same unknown distribution $P(X, Y)$, and the collection of data points is independently and identically distributed (i.i.d). In other words, changes between training and test sets are not tolerated. This assumption is very restrictive and commonly it is violated in real-world problems ([STORKEY, 2009](#)).

[Fawcett and Flach \(2005\)](#) have proposed a taxonomy to explain two types of problems: $X \rightarrow Y$, where the class y is defined by the covariate x , and $Y \rightarrow X$, where x depends on its class y , which is the covariate. Spam detection and medical diagnosis are well-known examples used in literature to illustrate both problems, respectively. In the former, an email is classified as spam or not-spam given its content. In the latter, medical diagnosis y defines which symptoms x will be seen.

For the sake of clarity, consider an instance x and a class y , the joint probability $P(x, y)$ is given by $P(x|y)P(y)$ in $Y \rightarrow X$ problems and $P(y|x)P(x)$ otherwise. Thus, dataset shift happens when changes in any of these elements occur between training and test sets. There are several dataset shifts according to which elements were affected:

- Covariate shift: $P(x)$ changes but $P(y|x)$ remains constant;
- Prior probability shift: $P(x|y)$ remains unaltered but $P(y)$ changes;
- Concept shift (or drift): $P(y|x)$ remains the same but $P(x)$ changes in $X \rightarrow Y$ problems or when $P(y)$ is unaltered but $P(x|y)$ changes in $Y \rightarrow X$.

In the case of quantification, changes in $P(y)$ are mandatory, otherwise, quantification learning would be useless and the best quantifier would be defined by the training classes

distribution. According to [González et al. \(2017a\)](#), some authors restrict quantification task to $Y \longrightarrow X$ problems while they argue that both problems are frequent in real-world applications. Although most of the quantification literature assumes that $P(x|y)$ remains constant, in this thesis we present some contributions under both scenarios.

1.2 Main contributions

The general objective of this thesis is to present improvements in binary quantification. Initially, our main proposal, named Data Stream Quantification by Score Inspection (SQSI), is an algorithm to tackle data streams that is resilient to changes in the feature space due to monitoring the scores provided by a classifier and detecting concept drifts in an unsupervised manner. When a change is detected, our first proposal requests the true labels of the last chunk and updates the quantifier, acting as a supervised approach. True labels are an expensive resource in many applications. Therefore, our next proposal, named SQSI with Instance Selection (SQSI-IS), reduces the dependency on actual labels even further when it detects a drift, applying a semi-supervised learning method. Our results show that our new proposal achieves similar results requiring 50% or less labeled examples.

Differently from classification algorithms for data streams, in quantification, the algorithms work on a stream composed of batches where drifts in the class distribution are expected. Actually, the quantification goal consists in predicting the class distribution of each batch. However, we noted that the drift tests available in the literature are sensitive to changes in the features as well as in the class attribute. It means that a significant difference in the class distribution will cause a drift flag. We advocate that an ideal detector for quantification task should either be insensitive to drifts in the class attribute or, preferably, explicit which is the type of change that occurred. For this reason, we propose the **Concept Distance Threshold (CDT)** algorithm, the first unsupervised drift detection method that explicitly targets to be insensitive to class imbalance. We have empirically shown that our proposal outperformed the most used strategy to flag concept drifts in an unsupervised setup.

After our first contributions in data streams quantification, we realized that quantification in traditional datasets has some unexplored questions. Firstly, we studied the HDy algorithm that is a state-of-the-art mixture model quantifier. We noted that HDy is an instance of a more general and unexplored framework. As a result, we formalize a framework that generalizes the Mixture Model approach for quantification. We name this framework DyS: Distribution y-Similarity. We make an extensive evaluation to rank the most suitable dissimilarity functions so that future work makes the best use of DyS. We empirically analyze the influence of the number of bins on quantification accuracy and provide recommendations for this parameter. Our results show that Hellinger Distance used in HDy ([GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013](#)), a state-of-the-art Mixture Model approach, is outperformed by other measures.

Quantification is not interested in predicting the class of each observation, but to measure the representativeness of each class in the set. The subtle difference between classification and quantification requires specific experimental designs to evaluate the performance of quantification methods. We argue that a critical variable, the size of the test sets, has been mostly ignored in the experimental comparison of counting methods. From our experiments, we observe that several approaches perform well for larger test sets. On the other hand, for tiny test sets, the quantifiers' performance is degraded and commonly is outperformed by a baseline strategy (Classify and Count). In this sense, we propose a more complete experimental design to evaluate quantifiers. We empirically demonstrate the relevance of the test set size to assess the performance of quantification methods. We show that current quantification methods generally have a mediocre performance on small test sets, and we propose a metalearning scheme to select the best quantifier based on the test set size that can outperform the best single quantification method.

Most of the quantification methods depend, to some degree, on a classifier in a preliminary step. Differently from classification, quantification lacks studies related to the influence of distinct conditions to which quantifiers can be submitted. We show that score overlap, which is related to the classification difficulty, can define which quantification algorithm is more adequate to each problem. To mitigate the influence of scorer quality, we propose a quantification method that accounts for changes in the distribution of scores. Our proposal is named DyS with Synthetic Scores (DySyn) and uses synthetic scores to simulate scores with different qualities and find the best match between simulated and test scores. We show that our proposal outperforms the state-of-the-art in a comprehensive empirical evaluation involving 11 methods and six datasets.

1.3 Thesis organization

This thesis is organized as a collection of papers or submitted manuscripts, summarizing the main contributions of this research. Thus, each chapter is a straightforward transcription of these papers, including text, figures, background, and results. To adapt to the thesis format, minor modifications were made.

All chapters are self-contained, *i.e.*, all necessary background, required definitions, and notations are included. The presentation order of the papers, in this thesis, is in accordance with the chronological evolution of this research.

In the next sections, we introduce the content of each chapter and briefly describe the relationship among them.

1.3.1 Chapter 2

Title: "*Combining instance selection and self-training to improve data stream quantification*". This chapter is based on a paper published in the Journal of the Brazilian Computer Society (MALETZKE; REIS; BATISTA, 2018), that is an extension of a former paper titled

"Quantification in Data Streams: Initial Results", published in the 2017 Brazilian Conference on Intelligent Systems (BRACIS) (MALETZKE; REIS; BATISTA, 2017).

Our initial effort was concentrated on joining two distinct machine learning subareas motivated by a real-world application. The first, an important learning task that was still poorly explored, named quantification. The second is data streams mining, which has attracted a considerable amount of interest from the community, involving issues such as modifications in prior probabilities of classes, in data distributions, and in the class conditional probabilities.

To make our point clearer, consider a concrete example of the insect surveillance application. Mechanical traps currently are the most used approach in insect surveillance. These traps attract and imprison insects, but require a human expert to manually identify and count the number of captured insects of interest. Such human-centered approach restricts the scale in which insect surveillance can be applied in practice. Recent research efforts have developed a sensor to automatically recognize the species of insects using data of the wing movement (SILVA *et al.*, 2015a; CHEN *et al.*, 2014). This sensor uses machine learning algorithms to classify the signals obtained from the sensor into the insect species. For our purposes, classification is just an intermediate step. The final objective is quantification, *i.e.*, counting the number of insects that belong to a certain species. These counts are the estimate of the local insect population.

We consider the task of quantification in a non-stationary data stream setting. In this setting, we can face not only drifts in the class distribution, but also in the feature space. These concept drifts may make the current concept outdated, requiring learning a new classifier. For this reason, we also include a mechanism to request a minimal number of labels, doing it wisely.

Therefore, our first research contributions are summarized in this work and consist of:

- We propose the first algorithm to deal with an unexplored task in data stream problems named quantification;
- We present two other simple algorithms to define the *baseline* and *topline* methods for data stream quantification;
- We upgrade our first algorithm to actively select a smaller subset to be labeled by an external oracle and when a change is detected over the stream. Using semi-supervised learning, we label the remaining examples in order to train a new model;
- Our results show that our extension achieves similar results, requiring 50% or less labeled examples.

1.3.2 Chapter 3

Title: "On the Need of Class Ratio Insensitive Drift Tests for Data Streams". This chapter is based on a paper published in the International Workshop on Learning with Imbalanced

Domains: Theory and Applications on the 2018 ECML/PKDD (MALETZKE; REIS; BATISTA, 2018).

Concept drift detection is an important step in data stream mining and, for this purpose, a drift test is commonly applied. The simplest test assumes that the true label of an examples is made available as soon as the classifier predicts the class of said example. However, such an assumption does not hold for a large number of applications. On the other hand, unsupervised tests detect changes in the data distribution to flag concept drifts while not requiring class labels after deployment. Two examples are the nonparametric incremental drifts tests proposed by Kifer, Ben-David and Gehrke (2004) and Reis *et al.* (2016). These approaches compare two samples: the first is the one used to train the classifier and the second is the latest data in the stream. The test indicates a concept drift if such samples come from different data distributions.

The main point of this chapter is that the unsupervised drift tests available in literature are sensitive to changes in the features as well as in the class attribute. Additionally, such tests do not even distinguish flags that were caused by changes in features from those cause by changes in the class attribute.

We adapt a recently proposed technique for learning with recurrent concepts (REIS *et al.*, 2018) as an unsupervised concept drift test.

- We present a method that accurately flags concept drift occurrences in the feature space. Our approach is the first unsupervised drift detection method to target at being insensible to class imbalance explicitly;
- Our proposal provides as a byproduct an estimate for the class distribution, which is used to dynamically adjust the decision threshold of the classifiers;
- We show that our test is more accurate than the state-of-the-art for recognizing concept drifts and, as a consequence, leads to more accurate classifiers.

1.3.3 Chapter 4

Title: "*DyS: a Framework for Mixture Models in Quantification*". This chapter is based on paper published in the Thirty-Third AAAI Conference on Artificial Intelligence (MALETZKE *et al.*, 2019).

Among quantification methods, the Mixture Models (MM) family stands out due to its accuracy, simplicity, and ability to operate with imbalanced training samples. HDy (GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013) is the state-of-the-art Mixture Models algorithm. We argue that HDy consists of an instance of a more general algorithm that remains informal with relevant parameters to be evaluated.

In this chapter, we formalize a more generic MM approach, named DyS that is a Mixture Model framework for quantification. We made an extensive evaluation of our proposal and several achievements were found such as:

- We propose a framework that generalizes the Mixture Model approach for quantification;
- We redefine the parameters' values, showing that Hellinger Distance is not the best distance for quantification using mixture models;
- We include in our framework an ordinal dissimilarity function that is less affected by the sparseness used to model each distribution;
- Finally, we shown that Hellinger Distance used in HDy, a state-of-the-art MM approach, is outperformed by other measures.

1.3.4 Chapter 5

Title: *"The importance of the test set size in quantification assessment"*. This chapter is an extended version of the manuscript elaborated by André Maletzke, Waqar Hassan, Denis dos Reis, and Gustavo E.A.P.A. Batista and submitted to SIAM Conference on Data Mining (SDM20) ([MALETZKE et al., 2020](#)).

The main contributions of this chapter are:

- We show that test set size is an important variable to evaluate the performance of quantification methods. The analysis of a newly-proposed algorithm is incomplete without analyzing its performance with different test set sizes;
- We point out that current sophisticated quantification methods generally have a mediocre performance on tiny test sets, frequently underperforming Classify and Count, a baseline method;
- We show that a strategy for choosing which quantifier should be adopted depending on the test sample can outperform the best quantification methods when all test set sizes are considered.

We explore those findings by proposing a simple metalearning approach that selects the quantification algorithm according to metafeatures of the test set, including its size. This simple proposal has outperformed all binary quantifiers evaluated. Moreover, we advocate that, for now on, quantification research papers should use our experimental setup proposed in this paper to evaluate their proposals and compare to the state-of-the-art.

1.3.5 Chapter 6

Title: *"Accurately Quantifying under Score Variability"*. This chapter is based on a manuscript by André Maletzke, Denis dos Reis, Waqar Hassan, and Gustavo E.A.P.A. Batista submitted to 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (MALETZKE *et al.*, 2020).

Although classification and quantification are two different tasks, quantification frequently avails of classification methods as an intermediate step. Therefore, classification assumptions have a huge influence on quantification. In this sense, Tasche (2019) establish a connection between classification and quantification accuracy, showing that the score quality produced by a classifier has a significant impact over confidence intervals outputted by quantification methods. We also suspect this and thus investigate the impact that changes in the score quality, from training data to test data, can have in quantification accuracy. We hypothesize that changes in the data distribution can affect the score distribution produced by classifiers and consequently impact distinct quantification methods differently.

In this chapter, we show how binary quantifiers are affected when the quality of the scores outputted by classifiers vary. We also proposed a reliable quantifier under the score quality variability. This chapter describes the following contributions:

- We show the impact of the underlying classifier in the quantification performance;
- A method that accounts for score distribution changes in the quantification estimates.

1.3.6 Chapter 7

During the period of this research, we also collaborated to improve other areas, such as classification in data streams and one-class quantification. We contributed to collect a large dataset from different insect species, which was used by other collaborators in their researches. At last, we describe in this chapter the first experiment with an insect trap that captures insects selectively using a machine learning model.

1.3.7 Chapter 8

This chapter concludes our work, providing general considerations of all chapters. Chapter 8 also presents some future directions that emerged during the period of this research.

COMBINING INSTANCE SELECTION AND SELF-TRAINING TO IMPROVE DATA STREAM QUANTIFICATION

In the last years, learning from data streams has attracted the attention of researchers and practitioners due to its large number of applications. These applications have motivated the research community to propose a significant amount of methods to solve problems in diverse tasks, more prominently in classification, clustering, and anomaly detection. However, a relevant task known as quantification has remained mostly unexplored. The quantification goal is to provide an estimate of the class prevalence in an unlabeled set. Recently, we proposed the SQSI algorithm to quantify data streams with concept drifts. SQSI uses a statistical test to identify concept drifts and retrain the classifiers. However, the retraining involves requiring the labels for all newly arrived instances. In this chapter, we extend SQSI algorithm by exploring instance selection techniques allied to semi-supervised learning. The idea is to request the classes of a smaller subset of recent examples. Our experiments demonstrate that SQSI's extension significantly reduces the dependency on actual labels while maintaining or improving the quantification accuracy.

2.1 Introduction

As machine learning becomes a mature discipline, researchers and practitioners start considering more challenging application settings. One example is learning from data streams, in which data arrive continuously and may be non-stationary. Recently, data streams have attracted a considerable amount of interest from the machine learning community. Consequently, we have seen an increasing and diverse set of methods in several tasks such as classification, clustering, and anomaly detection ([HILL; MINSKER, 2010](#); [DYER; CAPO; POLIKAR, 2014](#); [FANAEE-T; GAMA, 2014](#); [NGUYEN; WOON; NG, 2015](#)).

However, a relevant task, named quantification, has remained mostly unexplored in this setting. The objective in quantification is to provide an accurate estimation of the class distribution of an unlabeled set (FORMAN, 2005). To elucidate the practical utility found in this task, consider as a concrete example the insect surveillance application described next.

Insect surveillance is the key to control agricultural pests and disease vectors. The knowledge of the location of the insects allows planning *local* interventions. Local interventions are critical in reducing costs and environmental impact. In contrast, global interventions, such as large-scale spraying of insecticides, frequently used in agriculture and public health, are expensive and associated with the contamination of soil and water, as well as with increasing danger of extinction for threatened species.

Mechanical traps are the most used approach in insect surveillance. These traps attract and imprison insects, but require a human expert to manually identify and count the number of captured insects of interest. Such human-centered approach restricts the scale of application for insect surveillance. Recent research efforts have developed a sensor to automatically recognize the species of insects using data from the wing movement (CHEN *et al.*, 2014; SILVA *et al.*, 2015b). This sensor uses machine learning algorithms to classify the signals obtained from the sensor into the insect species. For our purposes, classification is just an intermediate step. The final objective is quantification, *i.e.*, counting the number of insects that belong to a particular species. These counts are the estimates of the local insect population.

We can frame several additional applications as a quantification task. For instance, estimating the percentage of unemployed people across different time periods; counting the number of positive, negative, or neutral tweets about a specific topic or product; estimating the number of news related to terrorism in the last month or semester and many others.

The class distribution of a training set could be statically considered as an estimate of the class distribution of a test set, not requiring any form of learning. However, quantification problems typically implicate significant divergence between the class distributions of the training and test sets. For instance, news about a particular subject usually vary with time and accurately counting these articles requires a classification step.

According to Forman (2005), quantification is a supervised task, recently formalized as a machine learning problem. Quantification shares similarities with classification. For instance, both consider the same representation for examples and a nominal output feature describing the class. However, quantification is not particularly interested in predicting the label for each instance, but in the overall quantity of elements of a specific class. Consequently, a quantifier issues an output for a set of examples, instead of for each instance. The output consists of a sequence of real values which are the estimates of the classes distribution.

Quantification and classification are also related tasks that benefit from each other. Quantification can be straightforwardly achieved by counting the number of examples classified

as each class. This approach is known as *classify and count*. However, empirical evaluations, such as the one in (GAO; SEBASTIANI, 2016), have shown that *classify and count* usually leads to suboptimal results. Including additional information, such as the estimated error and classification scores, leads to more accurate results. Nonetheless, even elaborated approaches that make use of additional information usually maintain an underlying classification nonetheless. On the other hand, classification benefits from quantification as an estimate of the class distribution is required to perform calibration (VUK; CURK, 2006).

To the best of our knowledge, the quantification research has mainly focused on the batch setting (FORMAN, 2005; BELLA *et al.*, 2010; MILLI *et al.*, 2013a; GAO; SEBASTIANI, 2016). In such cases, excepting changes in the proportions of the classes, other aspects of the distribution of data are assumed to be stationary. Recently, we proposed an algorithm to estimate the class distribution in non-stationary data streams (MALETZKE; REIS; BATISTA, 2017). This algorithm, named Stream Quantification by Score Inspection (SQSI) is resilient to changes in the feature space due to it monitoring the scores provided by a classifier and detecting concept drifts in an unsupervised manner. However, when SQSI detects a drift, it requests the true labels of all events in the chunk and updates the quantifier accordingly, acting as a supervised approach.

True labels are an expensive resource in many applications. Therefore, in this chapter, we extend the SQSI algorithm with instance selection techniques to decide which instances will have their actual labels requested. In this new setting, instead of demanding class labels for all cases in the chunk, we actively select a smaller subset to be labeled by an external oracle. Afterward, we use semi-supervised learning to label the remaining examples in the chunk. Finally, we use the cases labeled by an oracle in conjunction with those labeled by a semi-supervised scheme to update our quantification model. Our objective is to reduce the SQSI dependency on actual labels even further when it detects a drift. Our results show that our extension achieves similar results to SQSI while typically requiring 50% or less labeled examples. We also demonstrate that cluster-based sampling is the instance selection method that provided the most consistent results across datasets.

The remaining of this chapter is organized as follows: first, we briefly present concepts related to data streams and quantification, as well as the instance selection and semi-supervised techniques explored in this chapter. Next, we describe the *topline* and *baseline* reference algorithms that were compared with our proposals. The original SQSI algorithm and our recent improvements are presented in sequence. Next, we show the experimental evaluation setup and results, respectively. Finally, we conclude this chapter with our final remarks and prospects for future work.

2.2 Background

In this section, we introduce relevant concepts regarding data streams and quantification, as well as instance selection strategies for semi-supervised learning.

2.2.1 Data stream

A data stream is an ordered sequence of instances. Let $E = (\vec{e}_1, \vec{e}_2, \dots, \vec{e}_t, \dots)$ be a data stream, where $\vec{e}_t \in \mathbb{R}^p$ is an instance in a p -dimensional feature space. In supervised problems, each instance $\vec{e}_t$ has an associated label. Thus, a supervised data stream can be represented by an ordered sequence of pairs $E_s = ((\vec{e}_1, y_1), (\vec{e}_2, y_2), \dots, (\vec{e}_t, y_t), \dots)$, where $y_t \in \{c_1, c_2, \dots, c_k\}$.

Real-world problems are increasingly modeled as data streams. This fact has motivated the development of novel methods that fulfill the requirements imposed by this type of data generation process, such as high volume, velocity, and volatility (NGUYEN; WOON; NG, 2015). Also, mining data stream faces other challenges that include identifying and reacting to changes in the feature space and updating the classification model to incorporate those changes (GAMA *et al.*, 2004). These changes in data, also known as concept drifts, are due to modifications in prior class probabilities $P(y_i)$, in data distributions $P(\vec{e}|y_i)$, and in class conditional probabilities $P(y_i|\vec{e})$. In general, two strategies are used to mitigate the concept drift impact: *i*) adapt the models at regular intervals and *ii*) use a trigger for concept drift presence and then adapt the model. There are several examples of methods that deal with concept drift in the literature (MASUD *et al.*, 2011a; REIS *et al.*, 2016). In our algorithm, we use the second approach and actively adapt a classification model when a drift is detected.

Another significant challenge in data stream mining is the occurrence of verification latency (MASUD *et al.*, 2011a). Most existing data stream classification algorithms assume that the true label of an instance becomes available as soon as these algorithms issue a classification. However, in real-world applications, this assumption rarely holds. Verification latency is the time delay before receiving the correct label (REIS *et al.*, 2016). Therefore, there usually is a delay of T_l units of time until the actual label becomes available. Many methods assume $T_l = 0$, and consequently, these methods can update their models using labeled events more rapidly than possible in practice. Unfortunately, in real situations, $T_l \gg 0$, and sometimes $T_l = \infty$, which makes the data stream mining even more challenging. Verification latency remains little explored in data stream mining, and only in the last years, some methods have been proposed (DYER; CAPO; POLIKAR, 2014; SOUZA *et al.*, 2015).

Verification latency occurs in most, if not all, quantification applications. If the true labels were known immediately or after a small amount of time, then it would be preferable to wait and count the true labels, instead of using a classification approach. In principle, most quantification applications should have significant or even extreme ($T_l = \infty$) verification latencies.

2.2.2 Quantification

Although quantification and classification share similarities, their objectives differ. Thus, quantification requires specific evaluation measures and, consequently, specialized machine learning algorithms (MILLI *et al.*, 2013a).

Given the labeled set $E = \{\vec{e}_1, \dots, \vec{e}_m\}$, where each event $\vec{e}_t$ has an associated label y_t , where $y_t \in C = \{c_1, c_2, \dots, c_k\}$, we want to learn a classifier δ defined by the function $\delta : E \rightarrow C$ that assigns a label c_i to each event $\vec{e}_t \in E$. The true frequency of a class c_i in E is defined by $freq_E(c_i) = \frac{|\{\vec{e}_t \in E | y_t = c_i\}|}{|E|}$, and the objective of a quantifier is to estimate $\widehat{freq}_E(c_i) \approx freq_E(c_i)$. These elements are sufficient to define the quantification problem (MILLI *et al.*, 2013a) as stated in the Definition 1.

Definition 1 (Quantification (MILLI *et al.*, 2013a)). Let T_r be a labeled training set defined by collection E , an unlabeled test set T_e , and a classifier δ learned using T_r . The quantification problem consists in finding the best estimation of class distribution in T_e , i.e., $\forall c_i \in C$ we want to minimize the difference between the real frequency $freq_{T_e}(c_i)$ and the estimated one $\widehat{freq}_{T_e}(c_i)$.

Most of the contributions regarding quantification algorithms require the output of a classifier in a preliminary step. These algorithms are named aggregative ones. On the other hand, non-aggregative algorithms are the methods that holistically provide a quantification estimate.

To the best of our knowledge, the literature has not addressed the problem of data stream quantification with concept drift and verification latency. We started dealing with this issue by exploiting fundamental methods for quantification. The next section presents these methods.

2.2.2.1 Classify and Count (CC)

As previously mentioned, a straightforward approach to quantification is known as *classify and count* and consists in labeling each instance using a classifier and then counting the number of examples in each class (GAO; SEBASTIANI, 2016). In other words, it estimates a class distribution by counting how many times each label was the output of a classifier δ for instances in a test set T_e , given that δ was induced with a training set T_r . We formalize this approach in Equation 2.1.

$$\hat{p}_{T_e}^{CC} = p_{T_e}(\hat{c}_i) = \frac{|\{\vec{e}_t \in T_e | \delta(\vec{e}_t) = c_i\}|}{|T_e|} \quad (2.1)$$

where $\hat{c}_i$ is the class assigned by the classifier, and $p_{T_e}(\hat{c}_i)$ is the fraction of instances in T_e assigned to c_i by the classifier.

In (GAO; SEBASTIANI, 2016), the authors argue that this strategy is suboptimal since it does not take into account other pieces of information such as the marginal errors among the classes. For example, consider two binary classifiers: h_1 with $FP = 20$ and $FN = 20$, and h_2

with $FP = 18$ and $FN = 20$ (FP and FN are the number of false positives and false negatives, respectively). In this scenario, the classifier h_2 is better than h_1 in terms of classification. However, for quantification, the classifier h_1 is the best binary quantifier, since the FP and FN are equal, and the errors cancel out each other, resulting in a perfect quantifier. Summing up, according to [Gao and Sebastiani \(2016\)](#), an inaccurate classifier can be an accurate quantifier if its marginal errors are spread as evenly as possible across FP and FN .

2.2.2.2 Probabilistic Classify and Count (PCC)

This approach computes $p_{T_e}(c_i)$ as an expected fraction of instances predicted to belong to class c_i ([GAO; SEBASTIANI, 2016](#)). Let $p(c_i|\vec{e}_j)$ be the probability of $\vec{e}_j$ belonging to the class c_i , estimated by the classifier, and $E[\vec{e}_j]$ the expected value of $\vec{e}_j$ defined in Equation 2.2.

$$\hat{p}_{T_e}^{PCC}(c_i) = E[p_{T_e}(\hat{c}_i)] = \frac{1}{|T_e|} \sum_{\vec{e}_t \in T_e} p(c_i|\vec{e}_t) \quad (2.2)$$

This method is a variation of CC and, in general, has shown good performance in comparison with other approaches ([GAO; SEBASTIANI, 2016](#)).

2.2.2.3 Adjusted Classify and Count (ACC)

This method estimates the actual proportion of a given class by applying a correction based on true positive and false positive rates (tpr and fpr , respectively) ([FORMAN, 2005](#)). Such rates are estimated with a validation set or with some cross-validation technique in T_r (e.g., leave-one-out approach). Equation 2.3 presents the calculation of ACC for a given class c_i .

$$\hat{p}_{T_e}^{ACC}(c_i) = \frac{\hat{p}_{T_e}^{CC}(c_i) - fpr_{T_r}}{tpr_{T_r} - fpr_{T_r}} \quad (2.3)$$

ACC accounts for the errors committed by its underlying classifier to adjust an estimation previously obtained with CC. If the estimations for the errors are perfect, so will be the estimate for the class proportions. However, it becomes harder to assure accurate estimates for those errors when the distribution of the data is non-stationary. Besides, a minor issue is that this method can produce negative results or results higher than 1. In these cases, a common practice is to clip these estimates.

Quantification also demands specific evaluation measures. We present the most used in the literature in the next section.

2.2.3 Evaluation metrics

Different measures can assess a quantifier accuracy. Most of them are based on error and cross-entropy measures ([FORMAN, 2005](#)). Quantification also inherits virtually any measures

that are suitable for regression problems. In this chapter, we use an intuitive measure based on absolute error. Equation 2.4 defines the Absolute Error (AE). It corresponds to the average absolute difference between the class distribution estimation, $\hat{p}(c_i)$, and the actual class distribution, $p(c_i)$ (GAO; SEBASTIANI, 2016).

$$AE(p, \hat{p}) = \frac{1}{|C|} \sum_{c_i \in C} |\hat{p}(c_i) - p(c_i)| \quad (2.4)$$

The AE ranges between 0 (best) and $\frac{2(1 - \min_{c_i \in C} p(c_i))}{|C|}$ (worst). Equation 2.5 presents the Normalized Absolute Error (NAE), normalized version that ranges between 0 and 1 (GAO; SEBASTIANI, 2016).

$$NAE(p, \hat{p}) = \frac{\sum_{c_i \in C} |\hat{p}(c_i) - p(c_i)|}{2(1 - \min_{c_i \in C} p(c_i))} \quad (2.5)$$

However, according to Gao and Sebastiani (2016) both AE and NAE may suffer from a serious issue when the true class prevalence is small. For instance, the same errors are produced for $p(c) = 0.01$ when predicting $\hat{p}(c) = 0.1$ and for $p(c) = 0.41$ when predicting $\hat{p}(c) = 0.50$. To avoid these cases, Gao and Sebastiani (2016) propose the Normalized Relative Absolute Error ($NRAE$) as in Equation 2.6.

$$NRAE(p, \hat{p}) = \frac{\sum_{c_i \in C} \frac{|\hat{p}(c_i) - p(c_i)|}{p(c_i)}}{|C| - 1 + \frac{1 - \min_{c_i \in C} p(c_i)}{\min_{c_i \in C} p(c_i)}} \quad (2.6)$$

$NRAE$ is undefined for zero denominators. Therefore, $p(c)$ and $\hat{p}(c)$ should be smoothed with Laplace Smoothing, as in Equation 2.7.

$$p_s(c) = \frac{\varepsilon + p(c)}{\varepsilon|C| + \sum_{c_i \in C} p(c_i)} \quad (2.7)$$

where $p_s(c)$ is the smoothed version of $p(c)$, and $\varepsilon = \frac{1}{2|E|}$ is the smoothing factor.

2.2.4 Instance selection and semi-supervised learning

The original SQSI approach (MALETZKE; REIS; BATISTA, 2017) splits the data stream into chunks with a non-overlapping sliding window, and processes each chunk in order. Whenever SQSI reports a concept drift, it induces a new classifier that is trained with the data in the chunk, requesting true labels for the all of its instances.

Although SQSI has the merit of requesting labels only when it detects a concept drift, labels are frequently a limited and expensive resource. Therefore, in this chapter, we propose the use of instance selection methods to find a subset of relevant examples in the chunk, and request

true labels only for them. We use a semi-supervised learning approach to expand these labels to the remaining examples in the chunk.

Instance selection methods are prevalent in the active learning literature. According to [Settles \(2010\)](#), there are three basics settings:

- Membership query synthesis: creates and labels a set of artificial instances;
- Pool-based: requests the true labels of a subset of the unlabeled examples;
- Stream-based selective sampling: the instances belong to a stream, and the decision to label instances happens in an online manner.

In our work, since we process data streams in chunks, we can avail of pool-based approaches. According to [Souza et al. \(2017\)](#), methods that select instances based on evaluation measures are widespread in the literature. These methods use approaches such as uncertainty, expected error reduction and query committee to estimate the utility of each example. Uncertainty sampling selects instances that the classifier is least confident about their class labels. Expected error reduction selects examples that contribute to reducing the model error. Query by committee use ensembles to decide which examples to label, such as instances with the largest vote discordance.

However, given a non-stationary scenario, where changes can happen in class distribution as well as in the feature space, the applicability of instance selection techniques based on evaluation measures may result in misleading selections. For example, the uncertainty sampling method selects the instances for which the model is least confident in its classification. Thus, this method indirectly assumes the data are stationary. When non-stationarity is present, an outdated model could be uncertain for all instances, or be confident in the wrong ones. For this reason, in our experiments, we only employ methods that do not use evaluation measures. We evaluate the following instance selection techniques.

2.2.4.1 Random Sampling

Random sampling consists in selecting a subset of the instances randomly and with equal probability ([ZLIOBAITE et al., 2014](#)). This technique can be considered naïve, since all instances are equiprobable and consequently no intelligent mechanism decides which ones are more valuable.

2.2.4.2 Farthest-First Traversal

The intuition behind this method is that a representative labeled set is composed of a set of diverse events. To achieve such diversity, the Farthest-First Traversal algorithm selects k instances that are the farthest from each other ([SOUZA et al., 2017](#)).

2.2.4.3 Cluster-based Sampling

Clustering analysis has been used in several areas to identify groups and observe the relationship between the groups in an unsupervised manner (ZLIOBAITE *et al.*, 2014; SOUZA *et al.*, 2017). There are plenty of clustering methods, but, in essence, most of them employ the idea of similarity. Thus, the clustering objective consists in maximizing the similarity between the instances in the same group and in minimizing the similarity between examples from different groups. Therefore, a pool of events represented by a cluster structure can be useful to pick out representative events. Instance selection based on clustering algorithms selects the events near to cluster centers and near to the cluster borders, to compose a diverse set of events that will have their labels requested. Finding clusters in a pool of events usually involves a parameter k that represents the number of clusters, and sometimes this parameter can be set according to the number of events that will be labeled. Another strategy consists in defining a number b of events for which the true labels will be requested, and a number be of border events from each cluster. Hence, the number of clusters is given by $k = \lfloor b/(be + 1) \rfloor$. Souza *et al.* (2017) uses this last strategy.

These instance selection methods attempt to overcome the labeling bottleneck by requesting an oracle to label only $b\%$ of the chunk. However, defining how many examples to label is not a trivial task. Therefore, we decided to evaluate a range of values. In fact, we are looking for a minimal number of labels that provide similar results to a full labeled set. However, sets with too few labeled instances may lead to an underfitted model, since the learning algorithm does not have enough instances to learn properly and has to extrapolate regardlessly. In any event, to optimize the use of such small labeled sets, we use semi-supervised methods to infer the labels of the remaining examples in the chunk.

Semi-supervised Learning (SSL) methods are useful for learning tasks in which the quantity of labeled examples is so limited that compromises the generalization ability of the learning algorithm. Zhu and Goldberg (2009) present a comprehensible survey of semi-supervised learning techniques. In our experimental section, we experiment with Self-training, one of the most straightforward SSL methods available.

Self-training is a wrapper algorithm that iteratively applies a supervised learning method. This algorithm induces an initial classification model using the labeled portion of the data, and uses this model to classify the unlabeled examples. In each iteration, part of the classified instances are moved to the labeled set. A popular approach is to move the instances classified with the highest confidence. The algorithm tag these instances with the predicted label. In our implementation, we decided to move one instance per iteration: the one with the highest score. The algorithm continues until it labels all examples. The Algorithm 1 details the Self-training strategy (SOUZA *et al.*, 2017).

According to Zhu and Goldberg (2009), Self-training assumes that its predictions, when

Algorithm 1 – Self-training**Input:** Labeled instances L_s ; Unlabeled instances U_s ; Supervised machine learning method ml_{alg} **Output:** $\delta(U_s)$

```

1: repeat
2:    $\delta \leftarrow \text{buildClassifier}(L_s, ml_{alg})$ 
3:    $S \leftarrow \text{SelectMostConfident}(\delta, U_s)$ 
4:    $U_s \leftarrow U_s - S$ 
5:    $L_s \leftarrow L_s \cup S$ 
6: until  $U_s = \emptyset$ 
   return  $\delta(U_s)$ 

```

performed with high confidence, tend to be correct. This assumption has a controversial point motivated by the fact that early errors made by the classifier δ can be propagated to the entire unlabeled set. For this reason, we reinforce the importance of the instance selection methods to choose the most representative instances to be externally labeled by an oracle.

2.3 Reference approaches for data stream quantification

This section presents two reference approaches to quantify data streams. Although these approaches make assumptions that we consider infeasible in practice, such as the absence of concept drifts and the instantaneous availability of all labels along the stream, we use them as reference methods.

2.3.1 Static

This algorithm ignores the non-stationarity present in data stream environments. It builds a quantifier with the first examples from the stream and does not update it over time. This approach generates a quantification every w events. Therefore, it requires only a small portion of labeled data from the beginning of the stream. It is also very efficient since it builds a quantifier only once. We consider it as a *baseline*.

2.3.2 Sliding

In this approach, we regularly update the quantifier every w events, attempting to track the most recent changes in the stream. After the quantification of the w events is estimated, their actual labels become available, allowing the quantifier to be updated. This setting represents a null-verification latency scenario. We consider this approach to be our *topline* reference.

In the next section, we present our proposal, and later compare it with the *baseline* and *topline* strategies.

2.4 The original and extended SQSI algorithm

In this section, we present the SQSI algorithm, proposed by [Maletzke, Reis and Batista \(2017\)](#), as well as the extended version that incorporates instance selection and semi-supervised learning.

The SQSI works as follows. As an initial step, it learns a classifier δ from a labeled training set. After this initialization, the method issues a quantification whenever a pool achieves w events. To this end, the SQSI generates classification scores for each event in the pool using the classifier δ . Then, we verify whether these scores and the ones estimated in the training set (calculated with cross-validation) come from the same distribution (using a statistical test), *i.e.*, we sense the presence or absence of a drift only between the training set scores and the scores of the pool of recent instances. If the null hypothesis is not rejected (*i.e.*, both samples come from the same distribution), we apply the quantification method, and the result is issued.

However, if the null hypothesis is rejected, we perform a linear transformation in the recent pool, applying Equation 2.8 ([REIS *et al.*, 2016](#)).

$$T(X_{f,i}^{act}) = \frac{X_{f,i}^{act} - \text{mean}(X_f^{act})}{sd(X_f^{act})} \times sd(X_f^{ref}) + \text{mean}(X_f^{ref}) \quad (2.8)$$

where X_f^{act} and X_f^{ref} are the set of the observed values for feature f in the recent (active) pool and the reference (training) set, respectively, and $X_{f,i}^{act}$ is the value of feature f of the i -th example in X_f^{act} .

The intuition behind Equation 2.8 is straightforward. The concept drift may be due to global linear transformation of the data. Therefore, SQSI first tries to reframe the attribute values in each example of the recent set with the mean and standard deviation of the reference set.

After applying the linear transformation, SQSI generates new scores for the transformed recent data pool. Afterward, it applies the statistical test to the new scores, and if the null hypothesis is rejected once again, the algorithm requests the labels of the events in the pool and updates the classifier δ . Otherwise, the quantification is performed and the result is issued. The statistical test used in our algorithm is the Kolmogorov-Smirnov test with a significance level of 0.001. Such low significance level is necessary to minimize the number of false-positives due to consecutive reapplications of the test.

The original version of the SQSI algorithm requests as many labels as necessary to learn during the stream. In the extreme case, it can request labels for every single example. This happens if every single chunk represents a non-linear transformation of the previous chunk. In our extension, we propose the introduction of a mechanism to request a smaller number of labels when a change is detected. We hypothesize that the use of an instance selection mechanism allied to semi-supervised learning can provide similar results with a smaller label dependency.

Figure 5 represents our proposed extension, named SQSI with Instance Selection (SQSI-IS). Similarly to the SQSI, our proposal requires an initial step that creates a quantifier with the first labeled chunk of the stream (training set). Thus, as result, we have our first quantifier Q . However, differently from SQSI, the SQSI-IS defines an instance selection method as well as the rate of the requested true labels.

In summary, the main difference between the SQSI and SQSI-IS consists in the use of a mechanism that combines instance selection and Self-training, when the linear transformation is insufficient to reframe the active pool, and therefore true labels need to be requested. This mechanism is displayed on items 1 and 2 of the Figure 5.

Figure 5 – Schematic design of the SQSI-IS algorithm.

Source: Elaborated by the author.

Aforementioned scheme, we present in Algorithm 2 our extension to the SQSI algorithm. The essence of the SQSI algorithm remains and the major difference between SQSI and SQSI-IS is in lines 14-16, where SQSI-IS requests a portion of the labels using an instance selection method and the Self-training algorithm labels the remaining examples.

2.5 Time complexity analysis

The time complexity of SQSI-IS comprises four factors, as follows: **1)** the complexity of performing the hypothesis test, **2)** the complexity of applying the linear transformation exhibited in Equation 2.8; **3)** the complexity of the instance selection method, and **4)** the complexity of training a new quantification model.

For the analysis, consider w the size of each batch, n the number of batches in the stream, N the number of observations $n \times w$, F the number of features, and γ the proportion of batches

Algorithm 2 – SQSI with instance selection and semi-supervised learning

Input: Initial training set X^{ref} ; Classifying algorithm ml_{alg} ; Data stream E ; Pool size w ; Quantification method qnt_{method} ; Instance selection method IS_{method}

Output: Class prevalence for each pool Qnt

```

1:  $\delta \leftarrow \text{buildClassifier}(X^{ref}, ml_{alg})$ 
2:  $X_{sc}^{ref} \leftarrow \text{getScores}(X^{ref}, ml_{alg})$ 
3:  $X^{act} \leftarrow \emptyset$ 
4: for each event  $\vec{e} \in E$  do
5:    $X^{act} \leftarrow X^{act} \cup \{\vec{e}\}$ 
6:   if  $|X^{act}| = w$  then
7:      $X_{sc}^{act} \leftarrow \delta(X^{act})$ 
8:      $t \leftarrow \text{Test\_KS}(X_{sc}^{ref}, X_{sc}^{act})$ 
9:     if  $t < 0.001$  then
10:       $X^{act} \leftarrow \text{transformData}(X^{ref}, X^{act})$ 
11:       $X_{sc}^{act} \leftarrow \delta(X^{act})$ 
12:       $t \leftarrow \text{Test\_KS}(X_{sc}^{ref}, X_{sc}^{act})$ 
13:      if  $t < 0.001$  then
14:         $S_L \leftarrow \text{requestLabels\_withInstanceSelection}(X^{act}, IS_{method})$ 
15:         $S_U \leftarrow X^{act} - S_L$ 
16:         $X^{ref} \leftarrow \text{Self-Training}(S_L, S_U, \text{One-NearestNeighbor})$ 
17:         $\delta \leftarrow \text{buildClassifier}(X^{ref}, ml_{alg})$ 
18:      end if
19:    end if
20:     $Qnt \leftarrow Qnt \cup qnt_{method}(\delta, X^{act})$ 
21:     $X^{act} \leftarrow \emptyset$ 
22:  end if
23: end for
    return  $Qnt$ 

```

that had true labels requested for.

For the first component, our hypothesis test of choice is the Kolmogorov-Smirnov test. Its computation is linear, given a list of ordered observations from both samples. As we need to order the observations, and as we apply the test for each every feature, its time complexity for each batch is $O(Fw \log w)$ and, for the whole stream, it is $O(nFw \log w)$, or $O(NF \log w)$.

For the second factor, the transformation exhibited in Equation 2.8 requires only the mean and the standard deviation of both samples, which can be obtained linearly. Therefore, the time complexity of the transformation is $O(w)$ for a single batch and $O(N)$ for the whole stream.

For the last two factors, as both base quantification model and instance selection technique are parameters, we note their time complexities as $(Q(x))$ and $(S(x))$, respectively. We only perform instance selection when training and as such, the complexity for training new model for a given batch is $O(Q(w) + S(w))$. However, the objective of SQSI is to lower the number of times we need to retrain the model. Then, the time complexity of retraining for the whole stream is $O(\gamma n(Q(w) + S(w)))$.

Finally, we can sum up the factors and we end up with the following time complexity for the whole stream: $O(n(\log(w) + \gamma(Q(w) + S(w))))$.

2.6 Experiments

Our experiments involve four real datasets that are described below:

- **Bike** (FANAEE-T; GAMA, 2014) contains hourly records of a bicycle-sharing system with the corresponding weather and seasonal information between years 2011 and 2012. The goal is to predict whether there is high or low demand. Thus, we expect concept drift due to seasonality. It contains 17,379 instances;
- **Mosquitoes** has laboratory data with information of mosquitoes passing through a photo-sensitive sensor. The sensor data comprise seven features for each event: the Wing-Beat Frequency (WBF) and the frequencies of the first six harmonics. The temperature varies during the stream, influencing the insect metabolism, and, consequently, changing their wing-beat frequency (TAYLOR, 1963; MELLANBY, 1936). We consider the temperature as a latent variable for the quantification and classification tasks. The objective is to distinguish between events of female *Aedes aegypti* and *Aedes albopictus* mosquitoes from *Culex quinquefasciatus* and *Anopheles aquasalis*. This dataset contains 13,410 instances;
- **Insects** contains events generated by the same insect sensor. Differently from the Mosquitoes dataset, the features are the wing-beat frequency and the 92 first coefficients of the frequency spectrum obtained with a 1024-point Fast Fourier Transform. The object is to differentiate the *Aedes aegypti* mosquitoes from the insects *Musca domestica*, *Culex quinquefasciatus*, *Culex tarsalis* and *Drosophila melanogaster*. It contains 83,339 instances;
- **NOAA** (DYER; CAPO; POLIKAR, 2014) composed of meteorological conditions registered by the U.S. National Oceanic and Atmospheric Administration (Bellueve-Nebraska) for 50 years. This dataset contains eight features and 18,159 daily registers;
- **Arabic-Digit** (HAMMAMI; BEDDA, 2010) A modified version of Arabic-Digit, described by a fixed number of MFCC values for the human speech of Arabic digits (among 10). The spoken digit defines the context, and the task is to predict the sex of the speaker. The dataset contains 26 features and 14,380 registers;
- **QG** (REIS; MALETZKE; BATISTA, 2018) is a version of the dataset Handwritten [10], constrained to the handwritten letters *g*, and *q*. The context is defined by the author (among 10), and the objective is to predict the letter. This dataset is composed of 63 features and 13,279 registers.

In our experiments, all the quantification algorithms for data stream belong to the aggregative class, *i.e.*, they require the classification of each event as an intermediate step. For this reason, we perform our experiments using two well-known classification algorithms that usually present very competitive results for a large number of application domains: Random Forest (RF) and Support Vector Machines (SVM). We used both with default settings available in WEKA tool. Therefore, we use both RF and SVM for all the quantification methods, *i.e.*, *Static*, *Sliding*, SQSI, and SQSI-IS.

To compare our results with (MALETZKE; REIS; BATISTA, 2017), we retain the same experimental setup, *i.e.*, we train an initial classifier with 400 examples, and the pool size has a fixed number of 300 examples. To analyze the impact of instance selection techniques, we evaluate the influence of different instance selection techniques in the SQSI-IS algorithm. We evaluate the following techniques: Random sampling, Farthest-First, and sampling based on clustering (*k*-Means). We vary the portion of selected events from 5% to 95% by increments of 5%.

Random sampling randomly selects $b\%$ instances with equal probability. Farthest-First selects an initial example at random with constant probability. This example is removed from the pool. The next example is the most distant one from the first. The process continues, removing the last chosen example and searching for the most distant example from the last instance. The process stops when it reaches $b\%$ instances.

For cluster-based sampling, we use the *k*-Means algorithm. We run the *k*-Means algorithm to find 2 clusters ($k = \lfloor b/(be + 1) \rfloor$). Thus, we request labels for the two events nearest to each cluster center and another eight true labels for the border events. In the experimental setup we consider $be = 3$, motivated by results presented in (SOUZA *et al.*, 2017).

Finally, after the labeling process, using one of the instance selection methods, we have two subsets of events. The first one has $b\%$ of labeled events and the second contains all remaining unlabeled examples. To avoid the risk of underfitting due to learning in an excessively small labeled sample, we apply the Self-training algorithm that iteratively labels the remaining events, using a One-Nearest Neighbor classifier (ZHU; GOLDBERG, 2009).

We execute all experiments ten times, and report averaged results across the ten runs. We assess our results with NRAE and compare the methods using the Friedman test with 95% confidence level and Nemenyi post-hoc test. Our evaluation is not a strictly prequential scheme, which means the tested examples are used for training only when a drift is detected. Additionally, for the SQSI-IS algorithm, the tested examples used for training are selected using an instance selection method. Implementation details and datasets are freely available as supplementary material (MALETZKE; REIS; BATISTA, 2018).

2.7 Results

In this section, we present the results of the SQSI-IS regarding the influence of instance selection strategies in data stream quantification. Our evaluation approach uses a sliding window of 300 examples. In addition, we train an initial model with the 400 first examples from the stream.

We evaluated three different instance selection techniques: Random, Farthest-First, and clustering-based sampling (k -Means). Our experiments are an extension of the results published in (MALETZKE; REIS; BATISTA, 2017). Therefore, we compare the results with the algorithms presented there, which include the *baseline*, the *topline* and the original version of the SQSI. We group the results by the combination of classification algorithm and quantification method.

Figures 6, 7, and 8 show the mean quantification errors for the Random Forest classifier applying distinct quantification methods and instance selection techniques for each dataset. The horizontal axis represents the percentage of examples labeled by the instance selection approach.

Figure 6 – Mean quantification errors. Normalized relative quantification error for each dataset using the Random Forest (RF) as a classifier and the Classify and Count (CC) quantification method, varying the percentage of true labels requested by each instance selection technique.

Source: Elaborated by the author.

Figure 7 – Mean quantification errors. Normalized relative quantification error for each dataset using the Random Forest (RF) as a classifier and the Probabilistic Classify and Count (PCC) quantification method, varying the percentage of true labels requested in each active learning technique.

Source: Elaborated by the author.

Figures 6, 7, and 8 show a decreasing error tendency, as the percentage of true labels increases. This decreasing tendency is common to several plots. In general, results with less than 50% labeled examples tend to have a higher variance than the ones labeled with more than 50% of the window.

Among the instance selection methods, clustering-based sampling performs best. Random and Farthest-First techniques change places as second-best depending on the dataset and counting method. These results corroborate with the results presented in batch classification problems (SOUZA *et al.*, 2017; SETTLES, 2010).

SQSI performs well on datasets Bike, Insects, NOAA, Arabic-Digit, and QG. For these datasets, there is a clear separation between the *topline* and *baseline* quantifiers and SQSI tends to perform closer to the *topline*. For Mosquitoes dataset, the performances of the *baseline* and *topline* are quite close to each other. This is an indication that no concept drifts are present or the occurrence of a significant amount of data chunks with a severely unbalanced class distribution.

Figure 8 – Mean quantification errors. Normalized relative quantification error for each dataset using the Random Forest (RF) as a classifier and the Adjusted Classify and Count (ACC) quantification method, varying the percentage of true labels requested in each active learning technique.

Source: Elaborated by the author.

The unbalanced class distribution in the data chunks for each dataset will be discussed next. For the NOAA dataset, the KS test correctly identifies the lack of concept drifts. In total, KS flagged zero concept drifts with the RF classifier during the entire stream. Therefore, SQSI and SQSI-IS variants perform constantly across the entire stream.

For the Mosquitoes dataset, SQSI performs slightly worse than the baseline. However, in this case, the KS test flagged a concept drift in 18% of the pools. We hypothesize that most of these drifts are changes in the class distribution rather than in the feature space. We note that the KS statistic is sensitive to changes in both class and covariate attributes. These class changes are likely to be due to the circadian rhythm of the insects (CHEN *et al.*, 2014). Additionally, a severe class distribution variability over some chunks of the streams has influenced the performance of the *topline* algorithm negatively. When a new chunk arrives, the *topline* algorithm makes a quantification and then receives the true labels for updating the quantification model. However, when the most recent chunk containing no drifts but a severely unbalanced class distribution, the *topline* efficiency goes down. Summing up, scenarios with no concept drift and with a

severe class imbalance within the chunks are more promising to the *baseline* than the *topline*, because the latter will carry little information regarding one of the classes over to the following chunk of the data, while the former will keep its original, and possibly well balanced amount of information for each class. This rationale also applies to the SQSI and SQSI-IS algorithms, but for this a high drift detection rate is mandatory. Figure 9 shows histograms regarding the frequency of data chunks over different positive class distributions for each dataset.

Figure 9 – Positive class distribution histograms. Histograms regarding the frequency of data chunks over different positive class distributions for each dataset. The horizontal traced line represents 20% of the positive class distribution.

Source: Elaborated by the author.

These histograms pointed out that for Mosquitoes and Insects datasets there is a large quantity of data chunks with a severely unbalanced class distribution along the stream. Unsurprisingly, algorithms that updated the model more often were more adversely affected. This is more prominent when there is only a small number of concept drifts (specifically in the feature space).

Another issue is that the changes in the class priors are causing the quantification method to require more labels than necessary. In particular, SQSI retrains the classifier with different class distributions. In principle, observing the quantification errors, this approach of tracking the most recent class distribution did not work well. Possibly because the initial classifier was trained on a larger training set and, therefore, had a better representation of the classes.

These results indicate that a possible path for future work is the development of a statistical test that is insensitive to changes in the class priors.

Regarding the quantification methods PCC performs slightly better than CC, and CC better than ACC. ACC relies on classifier error estimates to improve the quantification output. However, our experimental setup imposes two restrictions on ACC. The first one is that we train

Table 1 – Mean quantification error (NRAE) for each instance selection techniques using different quantifiers induced by RF classifier.

	CC	PCC	ACC	Mean
<i>k</i> -Means	0.0758	0.0670	0.0846	0.0758
Farthest-First	0.1053	0.0952	0.1102	0.1036
Random	0.1011	0.0867	0.0966	0.0948

Source: Elaborated by the author.

on a small samples and, therefore, it is more difficult to obtain reliable estimates. The second one is that, in the datasets with concept drifts, those estimates tend to become less reliable with time.

Summarizing the Figures 6, 7, 8, we show in Table 1 the mean NRAE about all datasets for each instance selection technique using different quantifiers applying RF as base classifier. Although the results showed lower errors when the *k*-Means was used in conjunction with the PCC quantifier, the errors for all setups are quite similar.

To analyze these results in depth, we performed the Friedman test with Nemenyi post hoc analysis to verify whether the differences among instance selection techniques have statistical difference. For this, we applied a comparison among *K*-Means, Farther-First, and Random techniques considering all range of positive class distributions, but separately for each quantification method. Table 2 presents the results, where p -value > 0.05 is represented by (*ns*), p -value ≤ 0.05 by (*), p -value ≤ 0.01 by (**), and p -value ≤ 0.001 by (***).

Table 2 – Statistical analysis. Results for Friedman test and Nemenyi post hoc test between the active learning techniques using RF classifier in each quantifier method.

	SQSI-IS+ <i>k</i> -Means			SQSI-IS+Farthest-First		
	CC	PCC	ACC	CC	PCC	ACC
SQSI-IS+Farthest-First	***	***	**	—	—	—
SQSI-IS+Random	<i>ns</i>	*	***	<i>ns</i>	<i>ns</i>	<i>ns</i>

Source: Elaborated by the author.

As expected, the statistical analysis shows that the clustering strategy (using *k*-Means algorithm) outperforms, with statistical significance, the other strategies with the quantifiers PCC and ACC. Regarding the CC quantifier, the *k*-Means strategy was statistically superior to the Farthest-First. However, we did not find statistical differences between Cluster-based sampling and Random strategy with the CC quantifier.

Figures 10, 11, and 12 present the results for the SVM classifier.

The results for the SVM classifier corroborate with the results for Random Forrest. Once again, CC and PCC produced the best quantifiers and ACC presented slightly worse results. Once again, the *topline* has presented a bad performance for the Mosquitoes dataset that is affected by unbalanced data chunks as shown in Figure 9.

Figure 10 – Mean quantification errors. Normalized relative quantification error for each dataset using the SVM as a classifier and the CC quantification method, varying the percentage of true labels requested in each active learning technique.

Source: Elaborated by the author.

Table 3 is the analogous of Table 1 for SVM, instead of RF. Again, the clustering strategy has presented better results on average, except for the CC quantifier. In this case, Farthest-First outperforms k -Means.

Table 3 – Mean quantification error (NRAE) for each instance selection techniques using different quantifiers induced by SVM classifier.

	CC	PCC	ACC	Mean
k -Means	0.1059	0.0758	0.0853	0.0890
Farthest-First	0.1000	0.0907	0.1164	0.1023
Random	0.1060	0.0953	0.0967	0.0993

Source: Elaborated by the author.

This fact is even more evident by looking the Table 4. The statistical analysis shows that the instance selection based on clustering (using k -Means) outperforms the others strategies, except for the CC quantifier.

Figure 11 – Mean quantification errors. Normalized relative quantification error for each dataset using the SVM as a classifier and the PCC quantification method, varying the percentage of true labels requested in each active learning technique.

Source: Elaborated by the author.

Table 4 – Statistical analysis. Results for Friedman test and Nemenyi post hoc test between the active learning techniques using SVM classifier in each quantifier method.

	SQSI-IS+ k -Means			SQSI-IS+Farthest-First		
	CC	PCC	ACC	CC	PCC	ACC
SQSI-IS+Farthest-First	***	***	***	—	—	—
SQSI-IS+Random	*	***	***	<i>ns</i>	<i>ns</i>	<i>ns</i>

Source: Elaborated by the author.

In fact, the Cluster-based sampling produces results with a lower variability compared to Random and the Farthest-First strategies. Furthermore, Cluster-based sampling frequently produces results similar to (or better than) the original SQSI using less than 50% of the true labels. Summing up, considering our datasets and the experimental setup, we point out the Cluster-based sampling as the most promising instance selection technique among the evaluated ones. Indeed, instance selection methods produced equivalent or even better results using only a portion of the true labels. This remark is valid for both base classifiers.

Figure 12 – Mean quantification errors. Normalized relative quantification error for each dataset using the SVM as a classifier and the ACC quantification method, varying the percentage of true labels requested in each active learning technique.

Source: Elaborated by the author.

Additionally, we note that our extension achieves similar results to SQSI typically requiring 50% or less labeled examples. Based on this observation, we perform a statistical analysis between SQSI and SQSI-IS allowing that only 50% of labels can be requested when a drift is detected for each instance selection technique. Table 5 summarizes this comparison, showing the NRAE of the SQSI for each quantifier at the table header and the NRAE of the SQSI-IS with the p -value for each instance selection technique in the rows.

The SQSI NRAE for all datasets using CC, PCC, and ACC quantifiers generated using a RF classifier were 0.0648, 0.0546, and 0.0599, respectively. Although the SQSI-IS NRAE values using different instance selection techniques have been slightly greater than the SQSI errors, there is no statistical difference.

In addition, when the quantifiers were induced using the SVM classifier, the differences between SQSI and SQSI-IS were greater. However, a statistical difference was not observed among all variations of SQSI-IS for all instance selection techniques. Adding a large number of datasets, the similarity between the SQSI and SQSI will be more evident. In general, our results

Table 5 – Quantification error and statistical analysis. Mean error and results for Friedman test between the SQSI and SQSI-IS using different instance selection techniques for each quantification method.

	SQSI with RF					
	CC (NRAE=0.0648)		PCC (NRAE=0.0546)		ACC (NRAE=0.0599)	
	NRAE	<i>p</i> -value	NRAE	<i>p</i> -value	NRAE	<i>p</i> -value
SQSI-IS+ <i>k</i> -Means	0.0758	<i>ns</i>	0.0670	<i>ns</i>	0.0846	<i>ns</i>
SQSI-IS+Farthest-First	0.1053	<i>ns</i>	0.0952	<i>ns</i>	0.1102	<i>ns</i>
SQSI-IS+Random	0.1011	<i>ns</i>	0.0867	<i>ns</i>	0.0966	<i>ns</i>
	SQSI with SVM					
	CC (NRAE=0.0574)		PCC (NRAE=0.0392)		ACC (NRAE=0.0504)	
	NRAE	<i>p</i> -value	NRAE	<i>p</i> -value	NRAE	<i>p</i> -value
SQSI-IS+ <i>k</i> -Means	0.1059	<i>ns</i>	0.0758	<i>ns</i>	0.0853	<i>ns</i>
SQSI-IS+Farthest-First	0.1000	<i>ns</i>	0.0907	<i>ns</i>	0.1165	<i>ns</i>
SQSI-IS+Random	0.1060	<i>ns</i>	0.0953	<i>ns</i>	0.0967	<i>ns</i>

Source: Elaborated by the author.

suggest that achieving a confident quantification rate in scenarios with labels scarcity is possible, especially when instance selection techniques are used.

Finally, results summarized in Table 5 are a response to our initial hypothesis showing that it is possible to require fewer labels when a drift is detected maintaining the performance.

2.8 Limitations

Our experimental setup has some limitations that are worth mentioning. The first limitation is that we only address binary quantification problems. We should note that although this limitation is present in our current setup, it is not a limitation to our proposal: SQSI-IS does not particularly rely on the underlying quantification problem being binary. Additionally, this limitation is shared with the majority of the quantification literature, as the most thought-through and best performing quantification techniques are designed exclusively for this setting.

A second limitation is that we simplified the experiments by analyzing consecutive batches of data, instead of providing a quantification for the last w instances after every single instance is observed. As w is a small number compared with the size of the stream, this simplification should not significantly alter any performance measurement. Yet, this limitation can be withdrawn by using the Incremental Kolmogorov-Smirnov (REIS *et al.*, 2016).

A third limitation is that for every drift detected that could not be successfully identified as a linear transformation of the data, we still have to collect true labels and retrain the model. Our proposals aim at decreasing the number of true labels that are requested, but we reckon that we could be able to avoid requesting any true labels at all in more situations than we do. For instance, we know for sure that in some of our test datasets, as Arabic-Digit, concepts are recurrent. A memory mechanism could be inserted in order to avoid requesting labels for concepts that were learned in the past.

The fourth noteworthy limitation is that, as we feed the Kolmogorov-Smirnov test with

the scores produced by the classifier, *i.e.*, its confidence on each individual instance being positive, the test ends up being sensitive to changes in the proportion of the classes. This is counter-productive since, while changes in the prior distributions can negatively affect classifiers, this type of change without changes in the feature space should not affect the performance of quantifiers. Therefore, raising drift flags and requesting an updated model would be unnecessary for our quantification objective. For future work, we aim at replacing either what is used to feed the KS test or the test itself.

2.9 Conclusions

In this chapter, we presented an extension of the SQSI algorithm that quantifies stream data using fewer true labels than the original version of the SQSI. Indeed, in most datasets, less than 50% of true labels were sufficient to maintain or even outperform the original results.

We showed that instance selection techniques, used in conjunction with the Self-training algorithm, can be an useful tool to tackle quantification problems on data stream scenarios where the labels are scarce.

We plan to investigate, as future work, different manners to look for events in each pool when true labels are required. For instance, favoring the selection of recent events in the pool, rather than old ones. Besides, other aspects that, for brevity, were limited in our experiments, such as the pool size and the training set size, will be explored and deeply analyzed in forthcoming studies.

ON THE NEED OF CLASS RATIO INSENSITIVE DRIFT TESTS FOR DATA STREAMS

Early approaches to detect concept drifts in data streams without actual class labels aim at minimizing external labeling costs. However, their functionality is dubious when presented with changes in the proportion of the classes over time, as such methods keep reporting concept drifts that would not damage the performance of the current classification model. In this chapter, we present an approach that can detect changes in the distribution of the features that is insensitive to changes in the distribution of the classes. The method also provides an estimate of the current class ratio and use it to adapt the threshold of a classification model trained with a balanced data. We show that the classification performance achieved by such a modified classifier is greater than that of a classifier trained with the same class distribution as the current imbalanced data.

3.1 Introduction

Class imbalance is an omnipresent problem. Researchers and practitioners have addressed the issue of learning from skewed data in a large number of application domains. In batch learning, an imbalanced dataset has one or more minority classes heavily outnumbered by the majority classes.

Such discrimination between minority and majority classes is not so clear for sequence data, as in online learning. In a data stream, the examples are not provided at once, but they arrive during the stream. Additionally, concept drifts can change the data distribution, requiring the online learner to update its model.

Concept drifts can occur in the features or the class attribute. Drifts in the class attribute may lead to imbalanced class ratios. Therefore, the relative frequency of the classes may depend on the span of the stream that is being processed. For this reason and differently from the batch setting, in online learning, a class can be both minority and majority within the same dataset.

To make this discussion more concrete, consider a real application of an insect surveillance sensor (SILVA *et al.*, 2015b). The objective of such a sensor is to predict the species of a flying insect using data obtained from the wings movement. The class distribution may be affected by several factors, including the circadian rhythm of the insects. The circadian rhythm is a biological process that governs peaks of activity and periods of resting. For many insect species, these peaks occur at dawn and dusk, as shown in Figure 13.

Figure 13 – Histograms representing the circadian rhythm of *Aedes aegypti* and *Culex quinquefasciatus* mosquitoes. At dawn, *Culex* is the dominant class, but it is outnumbered by *Aedes* at dusk.

Source: Reis *et al.* (2018).

A conventional approach to deal with concept drifts is to use drift tests. These tests can be divided into two categories: supervised and unsupervised. The difference lies in the need of class labels after deployment. The supervised tests assume that one examples' true class labels are available as soon as the classifier issues a prediction for this example. Such an assumption does not hold for a large number of applications. In the case of the insect sensor, the actual classes of the insects that passed in front of the sensor are expensive to obtain. In field conditions, this information could only be obtained by having an expert next to the sensor.

On the other hand, the unsupervised tests detect changes in the data distribution to flag concept drifts while not requiring class labels after deployment. Two examples are the nonparametric incremental drifts tests proposed by Kifer, Ben-David and Gehrke (2004) and Reis *et al.* (2016). These approaches compare two samples: the first is the one used to train the classifier and the second is the latest stream data. The test indicates a concept drift if these samples come from different data distributions.

The main point of this chapter is that the unsupervised drift tests available in the literature are sensitive to changes in the features as well as in the class attribute. It means that a significant difference in the class distribution will cause a drift flag. Moreover, such tests do not distinguish flags that were caused by changes in the features and the class attribute. However, we advocate that an ideal unsupervised detector should either be insensitive to drifts in the class attribute or,

Figure 14 – A comparison between two Random Forest classifiers on insect data. The classifier trained with a fixed and balanced class distribution outperforms the classifier trained with the same class distribution as the test set.

Source: Elaborated by the author.

preferably, explicit which is the type of change that occurred.

We show a simple precursory experiment to make our point clearer. Figure 14 shows a comparison between three *hypothetical* Random Forest classifiers assessed with three performance measures: AUC, F1, and accuracy. The horizontal axis represents the class distribution in the test set for a binary classification problem. The red and green curves represent classifiers that were trained with a fixed balanced training data. The first of them, however, has its threshold adjusted according to the expected class proportion at each test set¹, while the latter's threshold remains unchanged. The blue curve represents a classifier that is trained with data that share the same class distribution as each test set. Both classifiers trained with a fixed balanced class distribution performed better than the third classifier. This can be explained by the latter's need of extrapolating further with fewer examples for the minority class. Finally, among those classifiers trained upon a fixed balanced data, the one that has its threshold adjusted presented the best performance, which is expected.

From this experiment, we can advocate that if we have a classifier trained in a fairly balanced distribution, possibly with the aid of sampling techniques, there is no gain in retraining the classifier if the class distribution changes and the feature distribution remains the same, provided that the new class distribution is known. A better approach would be to keep the same classifier and adjust its decision threshold. Therefore, a concept drift test should preferably tell apart changes in the class distribution and changes in the distribution of the features.

In this chapter, we adapt a recently proposed technique for learning with recurrent

¹ The decision threshold is a scalar value that separates positive examples from negatives ones in a rank. Examples with a score greater than the threshold are labeled as positive. The remaining examples are marked as negative.

concepts (REIS *et al.*, 2018) as an unsupervised concept drift test. The proposed test is insensitive to changes in the class distribution and provides an estimate for the class distribution in the test set. In other words, the proposal is also a quantification method. We use such quantification estimate to adjust the decision threshold of the classifiers dynamically. We show that our test is more accurate than the state-of-the-art for recognizing concept drifts and, as a consequence, leads to more accurate classifiers.

This chapter is organized as follows: Section 3.2 summarizes the related work on data streams, concept drift, and quantification methods. Section 3.3 details the proposed method for detection of concept drifts. Section 3.4 explains the experimental setup and lists the datasets that are used in our evaluation. Section 3.5 analyzes our experimental results and compares them against the state-of-the-art. Finally, Section 3.6 presents our conclusions and directions for future work.

3.2 Related work

In this section, we summarize the relevant related work in concept drift detection and quantification in data streams.

3.2.1 Data streams and concept drift detection

A data stream is an ordered sequence of instances, $E = (\vec{e}_1, \vec{e}_2, \dots, \vec{e}_t, \dots)$, where $\vec{e}_t \in \mathbb{R}^p$ is an example in a p -dimensional feature space. In supervised problems, each instance $\vec{e}_t$ has an associated label. Thus, a supervised data stream can be represented by an ordered sequence of pairs $E_s = ((\vec{e}_1, y_1), (\vec{e}_2, y_2), \dots, (\vec{e}_t, y_t), \dots)$, where $y_t \in C = \{c_1, c_2, \dots, c_k\}$.

Data stream mining faces several challenges, such as high volume, velocity, and volatility (NGUYEN; WOON; NG, 2015). In several application domains, data distributions are nonstationary, leading to concept drifts. A concept drift is a significant change in the data distribution that may impact the classification performance of a system. Therefore, data streams require identifying and reacting to concept drifts as well as updating the models to incorporate those changes (GAMA *et al.*, 2004).

Supervised methods that deal with concept drifts assume the presence of actual labels throughout the entire stream. One example's true label is usually made available as soon as the model issues a prediction for this example. The true labels are used to retrain the model periodically or to detect drifts by monitoring the model's classification performance and retraining it when necessary (KUNCHEVA *et al.*, 2008; BIFET; GAVALDÀ, 2009; MASUD *et al.*, 2011b; WU; LI; HU, 2012). However, such supervised strategy strongly depends on the constant availability of labels to rebuild the model or to detect the concept drift. This dependence can rarely be fulfilled in real-world applications.

Some recent papers have explored the use of statistical tests as unsupervised triggers for concept drifts. Kifer, Ben-David and Gehrke (2004) applied hypothesis tests to detect concept drifts without labels. The hypothesis test verifies whether the distributions of training and test sets are similar. Žliobaitė (2010) has employed a similar strategy over data streams. A hypothesis test compares if two consecutive sliding windows present the same behavior. Maletzke, Reis and Batista (2017) and Reis *et al.* (2016) explore similar mechanisms, applying hypothesis tests to compare classification scores rather than feature values. The scores come from the training set and the most recent data stream sample. Such unsupervised detection approaches need current true labels only to retrain the model, similarly to an active learning approach, reducing the dependence on true label availability.

However, the statistical tests employed in these four papers are sensitive to changes in both features and the class attribute. More importantly, those changes are not distinguished. Therefore, if the only difference between two sets of data is the proportion of the classes, then those methods generically indicate a concept drift, regardless of the change in the class distribution not incurring the need of training a new classification model. In the previous section, we argued and showed empirical evidence that knowing how the class attribute changes suffice to adapt the threshold of a model and keep comparable or superior performance to a retrained model. Although classification performance can be affected by changes in the proportions of the classes, training a new classification model is not a better option than adapting the current model without requesting true labels. For this reason, detectors should not flag such changes as concept drifts but instead should provide by how much the class distribution changed.

In (REIS *et al.*, 2018) is proposed an approach to learn in the presence of recurrent *contexts*. The main assumption is that the data stream approximately follows a small set of data distributions, called context. Therefore, the authors propose two methods to identify the most similar context given the current data sample. One of these methods, the Single Most Relevant HDy (SMR-HDy), has a dual property: at the same time, it is a proxy for a distance between two data samples and a quantification method. We postpone the details of SMR-HDy to the next section. For now, let us clarify that, as a distance, SMR-HDy is not able to identify a new concept. Therefore, one of the contributions of this chapter is to give a statistical interpretation of SMR-HDy distances. We want to understand how likely a given distance is and flag a concept drift for distances beyond a statistical threshold.

As a quantification method, SMR-HDy provides an estimate of the positive class distribution. Our approach uses this estimate to adjust the decision threshold of a classifier. Additionally, in the process of estimating the positive class ratio, SMR-HDy does an internal search for the minimum distance between all assessed class ratios. We hypothesize that, due to this search, such distance is invariant to the class distributions and is sensitive just to feature drifts.

3.2.2 Quantification

Quantification is a supervised task, recently formalized as a machine learning problem (FORMAN, 2005). This task shares similarities with classification. For instance, both consider the same representation for examples and a nominal output feature describing the class. However, quantification is not particularly interested in predicting the label for each instance. Instead, it is interested in the overall quantity of elements of a specific class. Consequently, a quantifier issues an output for a set of examples, rather than one for each instance. The output is an estimate for the class distribution, which consists of a sequence of real values that are the estimates of the proportions of the individual classes in the set.

A straightforward, but usually inaccurate quantifier, is the Classify and Count (CC) approach. It consists of directly using a classifier to quantify a test set, *i.e.*, this method simply counts the number of examples predicted in each class. CC has a series of limitations. Although, it can provide optimal results for test sets with a class distribution that makes FPR equal to FNR², CC introduces a systemic error as the class distribution changes (GONZÁLEZ *et al.*, 2017a). This is a severe issue for quantification problems since quantification is only interesting when the class distribution is nonstationary. Otherwise, we could directly use the training set class distribution to predict the test set distribution.

The literature in quantification is quite vast, and we do not intend to do a comprehensive review in this chapter. We point the interested reader to (GONZÁLEZ *et al.*, 2017a) for a thorough survey on the subject. We are particularly interested in a class of methods known as Mixture Models (MM) (FORMAN, 2006), and one representative of this class named HDy (GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013).

The HDy builds two normalized (unit area) histograms, H^+ and H^- , for the scores obtained by the classifier on two validation sets with exclusively positive and exclusively negative instances, respectively. When presented with an unlabeled test set, the algorithm builds a histogram H^T with the scores obtained by the same classifier. These histograms, H^+ , H^- , and H^T , represent the distributions of the training set for each class and of the test set, respectively. Finally, HDy estimates the positive proportion rate as follows:

$$\text{HDy}(H^+, H^-, H^T) = \arg \min_{0 \leq \alpha \leq 1} \{ \text{HD}(\alpha H^+ + (1 - \alpha) H^-, H^T) \}$$

where HD is the Hellinger Distance (POLLARD, 2002), and each histogram, with B bins, is represented as a vector in the $\mathbb{R}^B$. Hellinger Distance is a function that estimates the dissimilarity between two probability distributions. Equation 3.1 defines Hellinger Distance for discrete

² FPR and FNR are the false positive and negative ratios, respectively.

distributions, as a histogram with B bins, and Figure 15 illustrates this process.

$$\text{HD}(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^B (\sqrt{p_i} - \sqrt{q_i})^2} \quad (3.1)$$

Figure 15 – HDy uses the Hellinger Distance (HD) to measure the dissimilarity between a mixture of positive and negative score distributions and an unlabelled score distribution, where α is the proportion of the positive class. HDy searches for an α that minimizes the Hellinger Distance.

Source: [Reis et al. \(2018\)](#).

The original intent of HDy was solely to estimate the proportion of the classes in a binary quantification problem. However, [Reis et al. \(2018\)](#) observe that HDy provides, as a byproduct, the distance between the test sample and the most similar linear combination of positive and negative training samples. Therefore, such byproduct distance can be taken as a dissimilarity measure between the training and test samples, while being insensitive to changes in the proportions of the classes.

SMR-HDy ([REIS et al., 2018](#)) assumes that a known set of concepts can approximately describe a data stream. Therefore, this approach stores the positive and negative training scores for all concepts and uses the byproduct distance of HDy to search for the most similar concept to the test sample.

The Hellinger Distance suits SMR-HDy since we can compare different distances and decide which one is smaller. In other words, we do not infer that a distance is either small or large, but instead, we verify which concept has the most similar data distribution to the current data by choosing the concept with the smallest HDy distance.

In this chapter, we do not assume the existence of a set of known concepts and develop SMR-HDy as a concept drift test. However, this distance lacks statistical meaning, and we are unable to affirm that it is big enough to infer that a concept drift has occurred with high probability. In the next section, we provide a statistical meaning for this single distance and offer a way of estimating a threshold that, once surpassed, indicates the occurrence of concept drift with high probability.

3.3 Proposal

As mentioned, the SMR-HDy algorithm consists in identifying, among a set of known concepts, which one is the most similar to the current stream data. In this chapter, we are not

interested in calculating the similarity to known concepts. Instead, we want to identify when the latest data significantly differs from the sample used to train the current classifier.

To this task, we need to set a threshold to the HDy distances so that distances greater than this value indicate a concept drift. We empirically observed that distributions of HDy distances are approximately normally distributed around a mean when the histograms H^+ , H^- , and H^T come from the same concept, even when the test set is imbalanced.

Figure 16 presents the results for ten benchmark binary datasets from UCI (DHEERU; TANISKIDOU, 2017) and OpenML (VANSCHOREN *et al.*, 2013) repositories. We uniformly and randomly split each dataset into two disjoint halves. The first half is reserved for training and validation, while the second for testing. To estimate the scores output by classification models, we used 10-fold cross-validation with the first half of the dataset. After obtaining the validation scores for H^+ and H^- , we induced classifiers in the first half.

Figure 16 – Distributions of the HDy distances without concept drifts.

Source: Elaborated by the author.

To make the best use of our limited data, we restricted the test set to 1,000 examples so that there would be enough examples to vary the proportion of the positive class from 0 to 1, and also provide variability across samples in the extremes of this range. We changed the positive class proportion from 0% to 100%, with increments of 1% at each time. For each positive ratio, we created 10 random samples of 1,000 events. No event appears more than once in each sample, even though it can appear on multiple samples.

Considering that the HDy distances are normally distributed, we can define an adequate threshold, measured in standard deviations from the mean, to flag when two samples A and B are significantly different. We propose a simple criterion to create a threshold (θ) based on the distributions of the distances. If the distance reported by HDy differs more than two standard deviations from the mean, we consider with high probability that there is a concept drift. Therefore, we propose a threshold θ as a decision boundary for whether a drift is reported.

According to our proposal, named **Concept Distance Threshold (CDT)**, θ is defined as follows:

$$\theta = \mu + 2\sigma \quad (3.2)$$

Subsequently, the decision D of whether a drift between two samples A and B is reported is defined as follows:

$$D(A, B) = \begin{cases} \text{drift,} & \text{HDy}(A, B) \geq \theta \\ \text{no drift,} & \text{otherwise} \end{cases}$$

We note that to the purpose of concept drift detection, only distances that significantly deviate from the mean on the upward side are reported, as they mean a greater measured distance between the data distributions than it is expected.

In the following section, we present our experimental setup and evaluation procedure to validate our proposal. We start the next section with a preliminary experiment to check the viability of our assumptions. Later on, we describe the four real datasets that were used to evaluate our method.

3.4 Experimental evaluation

We preliminarily evaluate our design decision with real-world data gathered with an insect sensor. According to literature, there is a broad set of factors that influence the flying behavior of insects (and, therefore, cause concept drifts) such as temperature, humidity, air pressure, age, availability of water and food and so on ([MALETZKE et al., 2018](#)). However, the temperature is a prominent factor. Figure 17 illustrates the influence of temperature on the wing-beat frequency, one of the features we extract from the signals.

The insect data were gathered in the laboratory using insectaries with sensors attached. Each insectary maintains insects of a single species and sex, leading to labeled data. We can also control temperature, humidity, and other ambient conditions maintaining the insectaries in climatized chambers.

We created a scenario to test our threshold. Let the ranges of temperature be the concepts and consider as current (or known) concept the data collected with under lower temperature ($\sim 24^\circ\text{C}$) and as new (or unknown) concept the data gathered under higher temperature ($\sim 34^\circ\text{C}$). We split the data described by the known concept ($\sim 24^\circ\text{C}$) into two disjoint halves, and we have repeated the same experimental design previously described (Section 3.3) to obtain the distribution of the distances. Afterward, we apply the classification model and get the scores for the positive and negative classes (H^+ and H^-). In the case of this experiment, the positive class

Figure 17 – Influence of temperature on the wing-beat frequency of female *Aedes aegypti* mosquitoes.

Source: [Reis et al. \(2018\)](#).

is composed of events of female *Aedes aegypti* and the negative class by male *Aedes aegypti* mosquitoes.

Figure 18 shows the distributions of the distances when the test set comes from the known and unknown concepts. There is a clear separation between the distances from different concepts, and we explore this difference to create a trigger to flag when data of an unknown concept arrives. The dashed line represents an upper threshold defined according to Equation 3.2. We observe that the distances computed between samples of the known concept remain mostly below the limit set by $\theta = \mu + 2\sigma$.

This preliminary evaluation indicates that our method is invariant to changes in the class distributions yet being sensitive to changes in the feature space. We further evaluate our proposal in the next experiments. For these, we use four real datasets:

Aedes-Culex A modified version of *Aedes aegypti*-*Culex quinquefasciatus* dataset, described by [Reis, Maletzke and Batista \(2018\)](#). This dataset contains 8,000 events described by features that register the passage of female *Aedes aegypti* and female *Culex quinquefasciatus* mosquitoes in front of an optical sensor. Besides wing beat frequency, there are other 25 numerical features obtained from the signal. Two ranges of temperature define the concepts, and the classification task is to distinguish between the two species. $W_{tr} = 2,000$ and $W_{ts} = 1,000$;

Aedes-Sex A portion of the dataset *Aedes aegypti*-sex, described in ([REIS; MALETZKE; BATISTA, 2018](#)) was used. This version contains only two concepts and 8,000 entries. Each event is described by the wing beat frequency, and other 25 numerical features

Figure 18 – Distance distribution obtained from test sets composed by examples from two different concepts. The known concept is insect data obtained at $\sim 24^\circ\text{C}$. The unknown concept was obtained at $\sim 34^\circ\text{C}$. The vertical line represents two standard deviations from the known concept mean.

Source: Elaborated by the author.

obtained from the passage of female and male *Aedes Aegypti* mosquitoes in front of the optical sensor. $W_{tr} = 2,000$ and $W_{ts} = 1,000$;

Arabic-Digit A modified version of the Arabic-Digit dataset (HAMMAMI; BEDDA, 2010; LICHMAN, 2013), which contains 8,800 entries described by a fixed number of MFCC values for the human speech of Arabic digits (among 10). The spoken digit defines the concept, and the task is to predict the sex of the speaker. $W_{tr} = 400$ and $W_{ts} = 200$;

Wine Wine Quality (CORTEZ *et al.*, 2009; LICHMAN, 2013) contains 6,498 entries described by 11 features. The classification task is to differentiate between two ranges of quality of wines. The concept is given by the type of the wine (red or white). $W_{tr} = 800$ and $W_{ts} = 300$.

Each dataset is divided into two parts of equal size. The examples of one of them intended exclusively for training a classification model and estimating a drift detection threshold according to CDT. The other part is intended to create test samples of size W_{ts} . Each part is composed of two well-known concepts. For each concept, from the training part, we select W_{tr} events with balanced class distribution to train the classification model used throughout our experimental evaluation. Every example in the same test sample belong to the same concept, and while an example does not appear more than once in the same sample, it can occur in different samples.

We select one of them to be the known concept (Tr_{kn}) and the other one to be the unknown concept (Tr_{un}). For each training set (Tr_{kn} and Tr_{un}), a classifier is induced (named δ_{kn} and δ_{un} , respectively). The rationale for using two training sets (Tr_{kn} and Tr_{un}) is straightforward: we want to evaluate at first the accuracy for detecting drifts. Based on the answer of our trigger,

we want to select which classifier is used for the classification task (δ_{kn} or δ_{un}), and evaluate the impact of the decision of the trigger on classification accuracy.

Additionally, we use the Tr_{kn} to learn the threshold (θ) according to the procedure described in Section 3.3. The only difference from that procedure is that we only use 100 instances to simulate test samples with different class distributions. This change is due to the limited number of entries in Tr_{kn} . Figure 19 (a) shows a schematic representation of the threshold learning process, and how the classifiers for each concept were created (δ_{kn} or δ_{un}).

Figure 19 – Experimental setup used to evaluate our proposal.

Source: Elaborated by the author.

Given an estimated threshold and the classifier for each concept, we measure the performance of our proposal regarding concept identification accuracy. We compare our proposal with an unsupervised method that applies a hypothesis test over the scores output by the classifier to identify drifts. For that, we use the Kolmogorov-Smirnov test (KS). This is a non-parametric test to compare distributions and is a state-of-the-art approach to identify concept drifts (KIFER; BEN-DAVID; GEHRKE, 2004; ŽLIOBAITĖ, 2010; REIS *et al.*, 2016; REIS; MALETZKE; BATISTA, 2018). We compare, using the KS test, the distributions of the scores obtained through cross-validation on the training set with output scores of the classifier δ_{kn} for each test sample. Similarly to (REIS *et al.*, 2016), we apply the KS test with a significance level of 0.001 to detect drifts.

To evaluate the capability of identifying when a concept drift occurs, we randomly extract test samples of size W_{ts} from the test set, according to Figure 19(b). For each sample, we vary the positive class proportion from 0% to 100%, with increments of 1%. For each positive ratio, we create 50 random samples of W_{ts} events (25 per concept). No event appears more than once in each sample, though it can occur on multiple samples.

Our proposal is suitable for binary classification and quantification problems. We assume that each processed chunk of data (referred to as sample) is entirely generated by only one

concept. Although the algorithm receives a sample as input, this chunk can represent a sliding window in a data stream.

All classification models are Random Forests with 200 trees, using `randomForest`³ package from R with default parameters.

In our experimental setup, a reliable concept drift trigger will only flag changes in the feature space and will ignore the simulated imbalanced class test sets.

3.5 Results

Table 6 presents the accuracy rates for concept drift detection obtained by CDT and KS algorithms, as well as the classification accuracies.

Table 6 – Mean accuracy rates for concept identification and classification. Standard deviations are in parentheses.

Datasets	Drift detection accuracy		Classification accuracy	
	CDT	KS	CDT	KS
Aedes-Culex	0.785 (0.239)	0.601 (0.192)	0.722 (0.158)	0.758 (0.151)
Aedes-Sex	1.000 (0.000)	0.567 (0.167)	0.982 (0.005)	0.802 (0.142)
Arabic-Digit	1.000 (0.000)	0.643 (0.220)	0.981 (0.005)	0.959 (0.019)
Wine	1.000 (0.000)	0.679 (0.222)	0.840 (0.061)	0.669 (0.143)

Source: Elaborated by the author.

The CDT algorithm performed consistently better than the KS algorithm regarding drifts detection. We note that CDT achieved perfect concept identification in three out of four datasets. Additionally, when our proposal was used as a trigger to detect concept drifts, we obtained better classification accuracy, except for the Aedes-Culex dataset. This indicates that for Aedes-Culex dataset the concept has no relevance for the classification task. On the other hand, for the Aedes-Sex, Arabic-Digit, and Wine datasets, the drift detection has led to the best accuracy rates with the lowest standard deviations.

As expected, the KS approach has shown to be severely affected by imbalanced class distributions in the test sets. For all datasets, the drift identification accuracy of the KS was lower than our proposal. This evaluation highlights the high sensibility of the KS test to the class distribution, restricting the KS test applicability. This result corroborates with the main point argued at the beginning, regarding the sensibility to changes in the class distribution of the existing unsupervised drifts tests.

Figure 20 (right) shows the sensitivity of the KS to class distribution. The drift detection accuracy is quite lower for extreme values of class distribution. Conversely, as shown in Figure 20

³ <<https://cran.r-project.org/web/packages/randomForest/index.html>>

(left), the CDT behavior seems to be insensitive regarding the class distribution, except to the Aedes-Culex dataset.

Figure 20 – Impact of class distribution over the drift detection accuracy. CDT (on the left) and KS (on the right).

Furthermore, regarding classification accuracy, we note an important relationship with the drift detection performance. For all datasets, except Aedes-Culex, the classification accuracy rates were higher when the CDT was used as a trigger. Figure 21 illustrates the classification accuracy for each positive class proportion.

For the Aedes-Culex dataset, the KS approach presented better classification accuracy even when the correct concept was not detected. We observed that the classifier δ_{un} performed well even on samples from the known concept. Contrarily, when the classifier δ_{kn} was applied to the samples from the unknown concept, the classification rate declined. Additionally, for this dataset, KS test chose the classifier δ_{un} for all samples, regardless of true concept. This fact explains the classification rate for the Aedes-Culex dataset.

Figure 21 – Impact of class distribution over the classification accuracy using CDT (on the left) and KS (on the right).

The CDT results corroborate with the initial assumption argued in Figure 14, except for the dataset Aedes-Culex, where our proposal has a limited drift detection accuracy. This can be

explained by the lower drift detection accuracy of our proposal for this dataset, selecting the wrong classifier mainly for the lower positive class distributions.

Two distinct aspects impact the CDT performance. First, our proposal has a high drift detection accuracy. For this reason, we use the adequate classifier for most of the test sets (δ_{kn} or δ_{un}). Second, the CDT provides the class distribution of each test set as a byproduct, and therefore we can use this information to adjust the decision threshold of the classifiers dynamically.

Our results suggest that in many real-world applications where concept drifts are expected, our proposal is the first method to detect drift with class distribution insensitivity.

Finally, our proposal has some limitations that are worth mentioning. The first limitation is that we only address binary problems. This limitation is related to the underlying method used in our proposal, the HDy algorithm. A second limitation refers to the threshold learning phase, performed to estimate the value of θ for each dataset. This phase may be time-consuming, and it is a mandatory step for our method. Conversely, given the estimated threshold our proposal has a straightforward and fast application over data streams.

3.6 Conclusion

We present a method that accurately flags concept drift occurrences in the feature space. Our approach is the first unsupervised drift detection method to target at being insensible to class imbalance explicitly. Additionally, our proposal provides as a byproduct an estimate for the class distribution, which is used to adjust the decision threshold of the classifiers dynamically. We have empirically shown that our proposal outperformed the most used strategy to flag concept drifts in an unsupervised setup.

CDT is limited to binary classification. One possibility of extending our proposal to multi-class problems consists in aggregating multiple binary models, as is done in the one-vs-rest approach.

DYS: A FRAMEWORK FOR MIXTURE MODELS IN QUANTIFICATION

Quantification is an expanding research topic in Machine Learning literature. While in classification we are interested in obtaining the class of individual observations, in quantification we want to estimate the total number of instances that belong to each class. This subtle difference allows the development of several algorithms that incur smaller and more consistent errors than counting the classes issued by a classifier. Among such new quantification methods, one particular family stands out due to its accuracy, simplicity, and ability to operate with imbalanced training samples: Mixture Models (MM). Despite these desirable traits, MM, as a class of algorithms, lacks a more in-depth understanding concerning the influence of internal parameters on its performance. In this chapter, we generalize MM with a base framework called DyS: Distribution y -Similarity. With this framework, we perform a thorough evaluation of the most critical design decisions of MM models. For instance, we assess 15 dissimilarity functions to compare histograms with varying numbers of bins from 2 to 110 and, for the first time, make a connection between quantification accuracy and test sample size, with experiments covering 24 public benchmark datasets. We conclude that, when tuned, Topsøe is the histogram distance function that consistently leads to smaller quantification errors and, therefore, is recommended to general use, notwithstanding Hellinger Distance's popularity. To rid MM models of the dependency on a choice for the number of histogram bins, we introduce two dissimilarity functions that can operate directly on observations. We show that SORD, one of such measures, presents performance that is slightly inferior to the tuned Topsøe, while not requiring the sensible parameterization of the number of bins.

4.1 Introduction

Quantification is a task that is similar to classification in the sense that we are provided with a training set with labeled observations, and a test sample with unlabeled examples. However, in quantification, our objective is to predict the proportion of instances that belong to each class, rather than the label of each observation. The literature has proposed several methods that yield smaller quantification errors than simply classifying individual examples and counting the issued classes. [González *et al.* \(2017a\)](#) present a comprehensive survey of the area. A relevant finding is the fact that the errors committed by quantification methods are more consistent than those obtained with classifying and counting, since the absolute quantification error of the latter grows linearly around a predicted proportion for which the quantification error is zero ([FORMAN, 2008](#)).

Better estimations for the proportion of the classes are important for applications where the interest is in analyzing tendencies and behaviors of groups of individuals rather than specific classifications. Examples are quality control for seminal material ([GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013](#)), estimation of insect population in a region ([REIS *et al.*, 2018](#)), and sentiment analysis in social media ([ESULI; SEBASTIANI; ABBASI, 2010](#)).

Among the quantification methods found in literature, a family of algorithms stands out due to its quantification accuracy, simplicity, and ability to work on imbalanced training samples: Mixture Models (MM) ([FORMAN, 2005](#)).

MM methods constitute a family of quantification approaches where the probability distribution of each class is modeled individually and learned from a training set. As a test sample contains data from all classes at different proportions, its distribution is a parametric mixture of the classes' individual distributions, where the parameters are the proportions of the classes. Hence, the MM methods search for the parameters of such a mixture and, consequently, estimate the proportions of the classes in the test sample.

MM methods commonly represent the distribution of the classes using histograms as an approximation of the discretized Probability Density Function (PDF) ([GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013](#)). The values stored in such histograms are scores provided by classifiers and relate to the probability of each observation belonging to the positive class. This approach has three main advantages. First, memory compactness, since histograms summarize the original data as multiple examples are aggregated into a single histogram bin. Second, easiness and low computational cost for mixing distributions, as we use vectors to represent histograms and these vectors can be interpolated inexpensively. Finally, simplicity and low computational cost for comparing distributions, since we can use any dissimilarity function that operates on a vectorial space.

However, there are significant disadvantages that need to be addressed. While histograms with a small number of bins incur a more compact vision of the original data, they also incur in

greater information loss. The lost information could otherwise be required to differentiate similar distributions. On the other hand, as MM methods typically resort to dissimilarity functions that compare aligned bins in isolation, as the Hellinger Distance (GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013), histograms with many bins incur spatial sparseness and demand more training and test observations to reasonably estimate the distributions. Finally, several distinct dissimilarity functions can be employed, although there is no consensus regarding which one is the most adequate.

In this chapter, we formalize a framework that generalizes the Mixture Model approach for quantification. We name this framework DyS: Distribution y-Similarity. We make an extensive evaluation to rank the most suitable dissimilarity functions so that future work makes the best use of DyS. We empirically analyze the influence of the number of bins on quantification accuracy and provide recommendations for this parameter.

Finally, we introduce two dissimilarity functions that are compatible with the general framework proposed by DyS, even though they operate directly over observations rather than histograms. By using one of such distances, we are not required to summarize the original data and therefore lose information. Such distances can potentially better differentiate similar distributions while being immune to the curse of dimensionality. Our results show that Hellinger Distance used in HDy (GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013), a state-of-the-art MM approach, is outperformed by other measures. This finding is even more evident when we tune the number of histogram bins. Moreover, we propose a parameter-free distance for quantification that provides smaller quantification errors than all tested histogram distances when using the previous standard number of bins, as suggested by (GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013), and remains competitive even after this parameter is tuned.

4.2 Related work

Quantification is a supervised Data Mining task that shares several similarities with classification. Both require the same feature-based representation for observations and a nominal output attribute describing the individual classes.

At first glance, classifying and counting seems to be a practical solution for quantification. However, research papers have shown that such a method generally produces poor quantification performance (FORMAN, 2006; GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013; GAO; SEBASTIANI, 2016; GONZÁLEZ *et al.*, 2017b), and systematically under or overestimates the classes proportion as the test set class distribution changes (FORMAN, 2008). These facts led the community to the proposal of several quantification methods. Due to lack of space, we refer readers to (GONZÁLEZ *et al.*, 2017a) for a comprehensive survey on the subject and (BEKKER; DAVIS, 2018) for newer approaches that rely only on labeled positive

observations.

Our purpose remains concentrated particularly in a class of methods known as Mixture Models – MM. MM methods represent the probability distribution of each class separately and model the test set distribution as a mixture of the classes’ individual distributions, where the parameters are the proportions of the classes. Hence, the MM methods search for the parameters of such a mixture and, consequently, find the proportions of the classes in the test sample.

However, datasets are generally multidimensional. Accurately estimating a multidimensional probability distribution requires an increasing number of observations as the number of dimensions goes up, since increasing the number of dimensions also increases sparseness. Thus, using all original dimensions from the dataset raises not only the cost of data acquisition for training but also sets an elevated minimum size for test samples. Literature provides a simple way to avoid both undesirable characteristics: the indirect use of a scorer that maps the observations from the feature-space to $\mathbb{R}$ (GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013; REIS *et al.*, 2018; REIS *et al.*, 2018).

Generally speaking, a scorer outputs a number that is proportional to the probability of an observation belonging to the positive class, and is often an integral part of several classifiers, as Naïve Bayes and Support Vector Machines. Additionally, any classifier can be turned into a scorer when they are part of an ensemble. For example, Random Forests produces a score that is the proportion of the votes favoring the positive class. Unbiased scores can be individually obtained for the positive and negative class through k -fold cross-validation within the training set: for each k validation portion of the data, a scorer is induced with all $k - 1$ other parts. Scores are obtained by applying such a scorer on the validation portion, and the scores given to positive and negative observations are kept apart (FORMAN, 2005; GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013; PÉREZ-GÁLLEGO *et al.*, 2019).

Scores are individual observations from a data distribution, and we need to estimate and represent such a distribution. A simple way of expressing a distribution that is also convenient for enabling a straightforward mixture of different distributions is the discretized Probability Density Function (PDF) (GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013). It consists of the aggregation of scores into normalized histograms with b bins, so that the sum of all bins equals one. These histograms can then be treated as vectors in the $\mathbb{R}^b$, and pairs of histograms can be mixed at varying degrees by performing a linear interpolation. This mixing approach means that individual observations are weighted instead of discarded, although they lose detail after being categorized into bins. Finally, any dissimilarity function that operates on a vector space can be applied to compare pairs of distributions. An equivalent approach for representing distributions that is out of the scope of this chapter is to use the Cumulative Distribution Function (CDF) instead of PDF. Forman (2005) used this representation to propose the earliest MM method for quantification.

More recently, González-Castro, Alaiz-Rodríguez and Alegre (2013) proposed the HDy

algorithm. The method uses two normalized histograms (the normalization causes the sum of the bins to be one), P^+ and P^- that summarize the samples of scores S^+ and S^- . Such samples are obtained from two (possibly cross-validated) validation sets with exclusively positive and exclusively negative observations, respectively. When presented with an unlabeled test sample, the algorithm builds a histogram Q with the set of scores Z obtained by the same scorer on such a sample. These histograms, P^+ , P^- , and Q , represent the distributions of the training set for each class and the distribution of the test sample, respectively. Finally, given the histograms P^+ , P^- , and Q the $\text{HDy}(P^+, P^-, Q)$ estimates the positive proportion rate as

$$\text{HDy}(P^+, P^-, Q) = \arg \min_{0 \leq \alpha \leq 1} \{ \text{HD}(\alpha P^+ + (1 - \alpha) P^-, Q) \} \quad (4.1)$$

where HD is the Hellinger Distance ([POLLARD, 2002](#)), and each histogram, with b bins, is represented as a vector in the $\mathbb{R}^b$. Hellinger Distance is a function that estimates the similarity between two probability distributions P and Q , where $P = \alpha P^+ + (1 - \alpha) P^-$. The HDy authors use different numbers of bins from 10 to 110, with increments of 10, and the final proportion of positive labels in the test sample is the median of these 11 estimates. To estimate the value of α inside the algorithm, the authors of the original paper perform a linear search within the range $[0, 1]$.

Research papers have extended or adapted the HDy algorithm for building, for example, ensembles models for quantification ([PÉREZ-GÁLLEGO *et al.*, 2019](#)), context identification methods ([REIS *et al.*, 2018](#)), and concept drift detection approaches ([MALETZKE *et al.*, 2018](#)). These achievements were obtained by promoting slight changes to the HDy setup. Additionally, [Reis *et al.* \(2018\)](#) show that estimating α through Ternary Search makes HDy more efficient and generally more precise than through linear search, since local minima are very close to the global minimum. In this chapter, we argue that HDy consists of an instance of a more general algorithm that remains informal with relevant parameters to be evaluated.

4.3 DyS

We propose DyS, a generic framework for quantification based on the similarity of score distributions. Equation 4.2 formalizes our proposal. We note that DyS is a generalization of HDy.

$$\text{DyS}(S^+, S^-, Z) = \arg \min_{0 \leq \alpha \leq 1} \{ \text{DS}(\alpha H(S^+) + (1 - \alpha) H(S^-), H(Z)) \} \quad (4.2)$$

where DS is a dissimilarity function, and S^+ , S^- , Z are positive training sample, negative training sample, and test sample, respectively, and H is a function that converts a sample of scores into a representation that is compatible with DS and that supports mixing two distributions according to a factor α . For all tested histogram distances, H produces a histogram from the given sample, so

that we obtain P^+ from S^+ , P^- from S^- and Q from Z . As we discuss later in this chapter, for the proposed dissimilarity functions SORD and MKS, $H(x) = x$, since the proposed methods operate directly over samples of scores rather than binned histograms. Moreover, while $\alpha S^+ + (1 - \alpha) S^-$ is a simplified notation of the interpolation process that mixes two samples, it is accurate only for histograms. How the mixture is performed in SORD and KS is better described later.

4.3.1 Histogram dissimilarity

We analyze the impact of switching the dissimilarity function across a plurality of measures from the literature that is suitable for comparing histograms.

In this section, we list all compared functions, and point the interested reader to (CHA, 2008) for a thorough survey on these measures. In our work, we have selected the following dissimilarity functions: Squared Euclidean (SE), Manhattan (MH), Probabilistic Symmetric (PS), Topsøe (TS), Jensen Difference (JD), Taneja (TN), Hellinger (HD), Dice (DC), Jaccard (JC), Chebyshev (CB), Inner Product (IP), Kumar-Hassebrook (HB), Cosine (CS), and Harmonic Mean (HM). All employed histogram distances, except for ORD, are described in Table 7, where P and Q are two normalized histograms of same length b .

Table 7 – Dissimilarity Functions.

SE	$\sum_{i=1}^b (P_i - Q_i)^2$
MH	$\sum_{i=1}^b P_i - Q_i $
PS	$2 \sum_{i=1}^b \frac{(P_i - Q_i)^2}{P_i + Q_i}$
TS	$\sum_{i=1}^b \left(P_i \ln \left(\frac{2P_i}{P_i + Q_i} \right) + Q_i \ln \left(\frac{2Q_i}{P_i + Q_i} \right) \right)$
JD	$\sum_{i=1}^b \left[\frac{P_i \ln P_i + Q_i \ln Q_i}{2} - \left(\frac{P_i + Q_i}{2} \right) \ln \left(\frac{P_i + Q_i}{2} \right) \right]$
TN	$\sum_{i=1}^b \left(\frac{P_i + Q_i}{2} \right) \ln \left(\frac{P_i + Q_i}{2\sqrt{P_i Q_i}} \right)$
HD	$2 \sqrt{1 - \sum_{i=1}^b \sqrt{P_i Q_i}}$
DC	$\frac{\sum_{i=1}^b (P_i - Q_i)^2}{\sum_{i=1}^b P_i^2 + \sum_{i=1}^b Q_i^2}$
JC	$\frac{\sum_{i=1}^b (P_i - Q_i)^2}{\sum_{i=1}^b P_i^2 + \sum_{i=1}^b Q_i^2 - \sum_{i=1}^b P_i Q_i}$
CB	$\max_i P_i - Q_i $
IP	$P \bullet Q \sum_{i=1}^b P_i Q_i$
HB	$\frac{\sum_{i=1}^b P_i Q_i}{\sum_{i=1}^b P_i^2 + \sum_{i=1}^b Q_i^2 - \sum_{i=1}^b P_i Q_i}$
CS	$\frac{\sum_{i=1}^b P_i Q_i}{\sqrt{\sum_{i=1}^b P_i^2} \sqrt{\sum_{i=1}^b Q_i^2}}$
HM	$2 \sum_{i=1}^b \frac{P_i Q_i}{P_i + Q_i}$

Source: Adapted from Cha (2008).

Although Cha (2008) surveys a larger set of distances, we decided to exclude from our analysis all monotonic transformations of these distances since they are deemed to produce the same quantification estimates. This effect happens as, in DyS, we only search for the mixing parameter that produces the lowest dissimilarity value and disregard the value itself. Monotonic

transformations do not change the order of the values, and therefore we should find the same parameters. We also eliminated common but asymmetric functions so that we need not impose an arbitrary order between the mixed training distribution and the test distribution.

Apart from the dissimilarity functions listed in Table 7, we also tested the use of ORD (CHA; SRIHARI, 2002). To better understand such a distance, we note that in its original description, the histograms are normalized by multiplying every bin in P by $|Q|$ and every bin in Q by $|P|$. As such, each histogram is the allocation of $|P| \times |Q|$ units into b bins. The objective of ORD is to find the least number of movements that need be done to transform Q into P , where one movement is the transference of a unit from a bin to a *neighbor* bin. There is always a possible transformation since both normalized histograms have the same number of units. This setting happens to be a univariate case of the *minimum difference of pair assignment* (MDPA) of two distributions, which is a special case of the Earth Mover’s Distance (EMD) (RUBNER; TOMASI; GUIBAS, 1998). For this particular case, the proposers of ORD introduce a greedy algorithm that can compute the distance in $O(b)$, whereas the algorithms that are used to solve generic instances of EMD have higher time complexity. We note that the proposed algorithm also works when the histograms are normalized to have sum one, instead of $|P| \times |Q|$. The Algorithm 3 describes the rationale of such distance.

Algorithm 3 – Ordinal Distance

Input: Histograms to be compared P and Q

Output: Dissimilarity between P and Q

```

1: diffsum  $\leftarrow$  0
2: total_cost  $\leftarrow$  0
3: for  $i \leftarrow 1, \text{length}(P)$  do
4:   diffsum  $\leftarrow$  diffsum + ( $P[i] - Q[i]$ ) total_cost  $\leftarrow$  total_cost + |diffsum|
5: end for
   return total_cost

```

All histogram distances tested in our experiments, except for ORD, use each bin position in isolation. In other words, for each bin in one histogram, only one bin in the other histogram can affect the distance. For this reason, we believe that ORD is less susceptible to the curse of dimensionality and bad parameterization of the number of bins, when it is greater than the ideal. For all other distances, as we increase the number of bins and consequently their granularity, non-zeroed bins are more likely to contain fewer observations and be countered by zeroed bins in the opposing histogram. In fact, when the number of bins is infinity, only observations that are identical in both samples affect the distance and help decrease it. The distance is always one (the maximum value) when there are no identical observations. ORD, on the other hand, is less affected by an increasing number of bins, since the difference between mismatching bins is accounted for and any pair of bins (and non-identical observations) can affect the distance.

We introduce two dissimilarity functions that are suitable for DyS and do not rely on histograms, as they are able to operate directly on the observations from training and test samples.

Such distances also carry the same benefits from the histograms approach, allowing for mixing pairs of distributions to be compared with a third distribution. However, they have the additional benefits of being immune to the curse of dimensionality while not losing information since they do not simplify the original data. We explain these distances in the following sections.

4.3.2 Mixable Kolmogorov Smirnov

The first distance is the Mixable Kolmogorov Smirnov (MKS) statistic. It is an adaptation of the Kolmogorov Smirnov (KS) ([KOLMOGOROV, 1933](#)) statistic to compare two discrete empirical distributions and accounts for the first distribution being a weighted mixture of two distributions.

Equation 4.3 formalizes such a dissimilarity, where S^+ and S^- are the two samples that will be mixed together, according to the weights α and $1 - \alpha$, respectively, and Z is the third sample that represents the distribution to be compared with.

$$D_{MKS}(S^+, S^-, \alpha, Z) = \sup_x |\alpha F_{S^+}(x) + (1 - \alpha) F_{S^-}(x) - F_Z(x)| \quad (4.3)$$

where $F_Y(x)$ is the proportion of the observations in Y that are lower or equal than x .

4.3.3 SORD

The second dissimilarity function is the Sample ORD (SORD). SORD can be viewed as a special case of ORD where the number of bins is infinity: while ORD is the minimum cost necessary to transform a histogram into another one, SORD is the minimum cost to transform a sample into another one.

If we are using SORD simply as a measurement of the difference between two samples S and Z , every observation x is weighted as $w(x)$, defined as follows.

$$w(x) \begin{cases} |S|^{-1}, x \in S \\ |Z|^{-1}, x \in Z \end{cases}$$

This way, both samples share the same total weight and the transformation is feasible. The cost of transforming a fraction $f_{i,j}, 0 \leq f_{i,j} \leq 1$, of the i -th observation in S into the j -th observation in Z is $c(i, j) = |f_{i,j} w(S_i) S_i - w(Z_j) Z_j|$. The objective of SORD is therefore the

following optimization problem.

$$\begin{aligned}
& \underset{f_{i,j} \forall i,j}{\text{minimize}} \sum_i^{|S|} \sum_j^{|Z|} c(i,j) \\
& \text{subject to } \sum_j^{|Z|} f_{i,j} = 1 \quad \forall i \\
& w(Z_j)Z_j = w(S_i)S_i \sum_i^{|S|} f_{i,j} \quad \forall j
\end{aligned} \tag{4.4}$$

For the purpose of quantification, one of the samples compared by SORD is a mixture of two other samples: one that contains positive training observations and another with negative training observations. This mixed sample is compared to a test sample. In this scenario, we have to adjust the weights of the observations in the mixed sample: positive observations share the same weight, proportional to α , and the negative ones share the same weight, inversely proportional to α .

SORD can be efficiently computed in $O(|S \cup Z| \log |S \cup Z|)$ with a greedy approach, where S and Z are the two samples being compared. Algorithm 4 fully describes the distance computation with the necessary change to the weights when S is a mixture (with parameter α) of two samples (S^+ and S^-).

Algorithm 4 – SORD Dissimilarity Function

Input: Mixing samples S^+, S^- , mixing factor α , comparing sample Z

Output: Dissimilarity between $\alpha S^+ + (1 - \alpha) S^-$ and Z

$$w'(x) \begin{cases} \alpha |S^+|^{-1}, x \in S^+ \\ (1 - \alpha) |S^-|^{-1}, x \in S^- \\ -|Z|^{-1}, x \in Z \end{cases}$$

```

1:  $v \leftarrow$  sorted array with  $\forall x \in S^+ \cup S^- \cup Z$ 
2:  $\text{acc} \leftarrow w'(v[1])$ 
3:  $\text{total\_cost} \leftarrow 0$ 
4: for  $i \leftarrow 2, \text{length}(v)$  do
5:    $\delta \leftarrow v[i] - v[i-1]$ 
6:    $\text{total\_cost} \leftarrow \text{total\_cost} + |\delta \times \text{acc}|$ 
7:    $\text{acc} \leftarrow \text{acc} + w'(v[i])$ 
8: end for
   return  $\text{total\_cost}$ 

```

4.4 Experimental setup

In this chapter, we make a comprehensible experimental evaluation divided into two parts.

First, we hypothesize the existence of a relationship between the size of the test sample and the number of histogram bins that lead to the smallest error. The original HDy paper reports the estimated distribution based on the median across the use of varying number of bins from 10 to 110, with increments of 10. While this particular range may provide good quantification errors when the test sample has a large number of observations, we want to verify if the more general DyS, which includes HDy, is significantly influenced by the number of bins, and how.

We note that until now, although the ideal number of bins has not been studied, a decision for this parameter may not be completely uninformed and can be based on important insights. Histograms with too many bins are negatively affected by two aspects. The first aspect is that if the sample size is not large enough, large histograms can become too sparse, each bin can have excessively low weight, and ultimately, the dissimilarity function can face the curse of dimensionality. Exception for this rule is the use of ORD. Notably, we note that ORD avoids being affected by such sparseness since the relation between neighbor dimensions is considered, rather than each dimension contributing in isolation to the magnitude of the distance. The second aspect is that a large number of bins has the implicit assumption of high precision for the scores. On the other hand, if there are too few bins, we may be unable to differentiate distributions.

To verify the impact of the number of the bins, in all experiments, we vary the number of bins from 2 to 20 with increments of 2, and from 20 to 110 with increments of 10. The test sample size, on the other hand, varies from 10 to 100 with increments of 10 examples, and from 100 to 500 with increments of 100 examples.

Once we figure a satisfactory range for the number of bins for each dissimilarity function, we proceed to the second part of our evaluation. We consider a satisfactory range to be one that minimizes the largest number of bins necessary to obtain the smallest quantification errors for at least 95% of the cases. We analyze the impact of using different histogram distances in the DyS framework for binary quantification. With a fixed range of bins for each distance, we rank them according to the median quantification error produced by their use so that the top-ranked distances are those which lead to the smallest errors in each one of our experiments. We vary the test sample size from 10 to 500 in the aforementioned way. We analyze the behavior of the ranks for each dissimilarity function with a box plot.

For all experiments, we vary the positive class proportion from 0% to 100% with increments of 1%, and for each proportion, we execute 10 runs with different test samples.

We performed preliminary experiments and concluded that Ternary Search (TSearch) suits all tested dissimilarity functions. For this reason, it is used for all of our experiments. We note that the α that minimizes Squared Euclidean distance can be algebraically deduced in $O(1)$. However, we also use TSearch for this distance to maintain experimental consistency across distances.

In the next section, we enumerate and describe the datasets used in our experiments.

4.4.1 Datasets

Each dataset was uniformly split into two halves: training and test. With the training half, we performed 10-fold cross-validation to obtain the training scores used by DyS. The full training half was also used to train a single scorer that was applied on the test half to get a test score set. Test samples were sampled from the test score set according to the settings described in the previous section regarding class proportion and size. One observation does not appear more than once in a single test sample, although it can appear in more than one test sample. This procedure was used to make the best use of our limited data.

As the time complexity of both SORD and MKS grows linearithmic for the total number of observations involved (including training and test), we undersampled training scores to up to 1,000 per class, when using these distances. Despite such reduction, we believe this number of observation scores can still carry more information than a histogram.

We produced all scores using Random Forests with 200 trees. Also, we assess the performance of the quantifiers in our experiments using the Mean Absolute Error (MAE) (SEBASTIANI, 2018). MAE is the average of absolute differences between true (p) and predicted ($\hat{p}$) quantifications for a set of classes C , as shown in Equation 4.5.

$$MAE(p, \hat{p}) = \frac{1}{|C|} \sum_{c \in C} |\hat{p}(c) - p(c)| \quad (4.5)$$

Table 8 presents a brief description of the datasets used in our experiments obtained from UCI (DHEERU; TANISKIDOU, 2017), OpenML (VANSCHOREN *et al.*, 2013), PROMISE (SHIRABAD; MENZIES, 2005), and Reis (REIS *et al.*, 2018) repositories. Specific citations are requested for Bank Marketing (MORO; CORTEZ; RITA, 2014), Credit Card (YEH; LIEN, 2009), HTRU2 (LYON *et al.*, 2016), Mozilla4 (KORU; ZHANG; LIU, 2007), Mushroom (LINCOFF, 1989), Nomao (CANDILLIER; LEMAIRE, 2012), and Occupancy Detection (CANDANEDO; FELDHEIM, 2016). Additionally, we note that Jock A. Blackard and Colorado State University preserve copyright over Covertypes.

Three observations about the datasets are due. First, Wine Type dataset is similar to Wine Quality. However, we want to differentiate between white and red wines, rather than the wine quality. Second, Covertypes-reduced is a stratified sample from Covertypes due to its considerable size. The abbreviated version is still large enough for our purposes. ArabicDigit is a preprocessed version of the original so that all examples have the same number of features (REIS *et al.*, 2018), and the objective is to predict the sex of the speaker rather than which digit is spoken. Finally, all the described datasets represent binary classification problems.

Table 8 – Datasets description.

Dataset	Size	Features	Repository
Anuran Calls	6,585	22	UCI
Bank Marketing	45,211	16	UCI
BNG (vote)	39,366	9	OpenML
Click Prediction	39,948	11	OpenML
CMC	1,473	9	UCI
Coverttype-reduced	8,715	54	UCI
Credit Card	30,000	23	UCI
EEG Eye State	14,980	14	OpenML
HTRU2	17,898	8	UCI
JM1	10,880	21	PROMISE
Letter Recognition	20,000	16	UCI
MAGIC Gamma	19,020	10	UCI
Mozilla4	15,545	5	OpenML
Mushroom	8,124	22	OpenML
Nomao	34,465	118	OpenML
Occupancy Detection	20,560	5	UCI
Phoneme	5,404	5	OpenML
Spambase	4,601	57	UCI
Wine Type	6,497	12	UCI
AedesSex	24,000	27	Reis
AedesQuinx	24,000	27	Reis
ArabicDigit	8,800	27	UCI
Handwritten-QG	4,014	63	Reis
Wine Quality	6,497	12	UCI

Source: Research data.

4.5 Experimental evaluation

In this section, we present and discuss our experimental results. One of our main questions regards a possible relationship between the number of bins and test sample size, to achieve the smallest quantification error. We are interested in knowing whether there is a test sample size for which it is better to use more or fewer bins than for another sample size, for the same distance. In Figure 22, we illustrate the Mean Absolute Error (MAE) across datasets while varying the sample size, using the distance function *Topsøe*, as it represents the general behavior of other distances.

We can form three observations. First, the error is lower for more significant test samples, which is expected, since we are provided with more information about the test distribution. Second, for the *Topsøe* distance function, a smaller number of bins generally leads to smaller errors across all assessed test sample sizes. Third, greater sample sizes are less negatively impacted by a higher number of bins. This is explained by the lower sparseness of the bins with more observations in a higher dimensionality.

Both observations hold for all tested distances, except Cosine, Harmonic Mean, Kumar-

Figure 22 – Mean absolute quantification error averaged for all datasets, obtained with DyS with Topsøe and varying test sample size and number of histogram bins.

Source: Elaborated by the author.

Hassebrook, Inner Product, and ORD. For the first four distances, errors are smaller for datasets with fewer observations, and a lower number of bins led to more significant errors. However, such distances performed very poorly: all of them led to errors greater than 70% on average, which is worse than a baseline that always predicts a positive class ratio of 50% and, consequently, obtains a maximum error of 50%. For this reason, Cosine, Harmonic Mean, Kumar-Hassebrook, and Inner Product will not be considered from now on.

On the other hand, ORD performed as well as the other distances while being almost invariant to the number of bins. This can be explained because each dimension is not used in isolation, which makes it less affected by sparseness. However, similarly to other distances, error decayed as test sample size increased. Figure 23 illustrates this finding. Our supplemental material website¹ contains figures for all other distances.

In Figure 24, we observe that most of the best quantification results were obtained within up to 20 bins for all considered distances, except for ORD. In Table 9, we detail this finding: we present the smallest upper limit for the number of bins that was necessary to constrain 90%, 95% and 100% of the smallest quantification errors obtained for each distance. Hellinger Distance produced 95% of its best quantification results within the range from 2 to 14 bins. This finding contradicts the arbitrary range of 10 to 110 bins used by HDy's original authors to calculate the median positive class ratio.

¹ <https://sites.google.com/site/andregustavom/research/dys>

Figure 23 – Mean absolute quantification error averaged for all datasets, obtained with DyS with ORD and varying test sample size and number of histogram bins.

Source: Elaborated by the author.

Figure 24 – Frequencies of the number of bins that produced the smallest quantification error for each distance function, in DyS.

Source: Elaborated by the author.

To rank the distances by quantification error, for each setting, we considered the median of the positive class ratio obtained with DyS while varying the number of bins. The number of bins ranges from 2 to the number of bins that are necessary to constrain 95% of the best results

Table 9 – Smallest upper bound for the number of histogram bins that encloses 90%, 95% and 100% of the smallest quantification errors produced by each distance function, in DyS.

Distance	90%	95%	100%
CB	4	4	10
DC	10	14	60
HD	12	14	30
JC	10	12	60
JD	16	20	50
MH	8	10	50
ORD	90	100	110
PS	18	30	70
SE	16	40	110
TN	14	18	100
TS	16	18	50

produced by the distance that was used, according to Table 9. The rankings are presented in Figure 25.

Figure 25 – Aggregation of several rank positions for different distances in DyS. Each quantification was predicted as a median of estimates obtained for different numbers of histogram bins. The range of bins was individually tuned for each distance. Test sample size varied from 10 to 500.

Source: Elaborated by the author.

We note that each distance had its range of bins tuned individually with the same datasets that were used for this comparison. Exceptions are SORD and MKS, which do not make use of this optimized parameter. This inserts bias into the comparison. On the other hand, if we had used the range from 10 to 110 bins, with increments of 10, as suggested by HDy's authors, we would obtain the ranking presented in Figure 26. In this scenario, the top five best distances are

the same. We note that ORD jumps to first place as it is less affected by large histograms, and Topsøe's error increases.

Figure 26 – Aggregation of several rank positions for different distances in DyS. Each quantification was predicted as a median of estimates obtained for different numbers of histogram bins. The range of bins was [10,110] for all distances. Test sample size varied from 10 to 500.

Source: Elaborated by the author.

For the remaining of our analysis, we consider the tuned range of bins again. We varied the test sample size from 10 to 100 with increments of 10 and from 100 to 500 with increments of 100. This difference of granularity provides small sample sizes (less than 100 observations) more weight than all bigger sizes combined since there are more test cases for the former case. We can see in Figure 27 that SORD performs only slightly worse than Topsøe for all sample sizes with a similar variance. However, we must keep in mind that SORD is a parameter-free algorithm, *i.e.*, its results were obtained without previous tuning. On the other hand, the Topsøe results are a consequence of a time-consuming process aimed at tuning the number of bins to be used on the same datasets. Finally, we also potentially limited the best performance that SORD could achieve since we limited the size of the samples, due to the algorithm's higher computational cost in comparison with the histogram distances.

Additionally, ORD also performs closely to Topsøe, after tuning, although not as close as SORD. This is true even though ORD fell from the second to the seventh rank position after the tuning process. However, this change of rank is due to an increase in the absolute performance of Topsøe, rather than a change in the performance of ORD. The latter is mostly unaffected by the rise in the number of bins (after a minimum at which the different distributions become discernible).

Figure 27 – Comparison between SORD and Topsøe for varying test sample size. The shaded area corresponds to the standard deviations from the measured points, and thinner curves set the limit of the shaded areas.

Source: Elaborated by the author.

Figure 28 – Comparison between ORD and Topsøe for varying test sample size. The shaded area corresponds to the standard deviations from the measured points, and thinner curves set the limit of the shaded areas.

Source: Elaborated by the author.

4.6 Conclusions and future work

In this chapter, we introduced DyS, a framework of Mixture Models for quantification. We analyzed the use of several histogram distances and concluded that Topsøe offers the smallest quantification errors across several datasets and test sample sizes when we tune the number of histogram bins. We experimentally found that the best range for the number of bins in DyS varies for each distance function. While ORD is mostly unimpaired by an incorrect setting this parameter, for the majority of the distance functions, a suitable superior limit was below 20. Particularly, histograms with 14 or fewer bins provide at least 95% of the best quantification results when using Hellinger Distance. This finding opposes the arbitrary range from 10 to 110 bins used by HDy's original paper. Finally, we introduced a new dissimilarity function, SORD, that operates over observations rather than histograms, while still being compatible with the framework provided by DyS.

SORD outperforms all distances when they do not have their parameters tuned. On the other hand, when we tune the parameters, our parameter-free algorithm is outperformed by the Topsøe, Probabilistic Symmetric, and Jensen Difference dissimilarity functions, respectively. However, even then, SORD presents better results than the HDy, the function currently used by the state-of-art MMs, and is only slightly outperformed by Topsøe. ORD falls a little behind SORD, and even behind HDy. However, we argue that its performance is still competitive to Topsøe's in practical terms and, as it is mostly unaffected by a wrong parameterization of the number of bins. ORD is a viable and more time-efficient alternative to SORD when the tuning process cannot be done or is unreliable.

In future work, we plan on evaluating the impact of varying quality of scores on mixture models for quantification. The score quality is related to the difficulty of a dataset for classification. We plan on evaluating whether the ideal number of bins for histogram distances vary according to the quality of the scores. Additionally, we intend on assessing situations where there is a mismatch between the quality of scores produced for the training and the test. This situation can happen as a result of concept drift and incur circumstances where test samples are easier or harder to classify than the training set. Finally, we plan on extending all comparisons to other families of learning algorithms, to provide better positioning for Mixture Models inside Quantification literature as a whole.

THE IMPORTANCE OF THE TEST SET SIZE IN QUANTIFICATION ASSESSMENT

Quantification is a Data Mining task similar to classification in the sense that it learns from a training set with labeled data. However, quantification is not interested in predicting the class of each observation, but rather measure the representativeness of each class in the set. The subtle difference between classification and quantification requires specific experimental designs to evaluate the performance of quantification methods. Consequently, the community has developed performance measures and experimental setups tailored to quantification tasks. Nonetheless, we argue that a critical variable, the size of the test sets, has still been mostly ignored in the experimental comparison of counting methods. Such disregard has three main detrimental effects. First, it implicitly assumes that quantification methods will perform equally well for different test set sizes. Second, it increases the risk of cherry-picking by selecting a test set size for which a particular proposal performs best. Finally, it disregards the importance of designing methods that are suitable for different test set sizes. In this chapter, we discuss these issues with the support of one of the broadest experimental evaluations ever performed in quantification literature, with three main outcomes. *(i)* We empirically demonstrate the importance of the test set size to assess the performance of quantification methods. *(ii)* We show that current quantification methods generally have a mediocre performance on the smallest test sets. *(iii)* We propose a metalearning scheme to select the best quantifier based on the test set size that can outperform the best single quantification method.

5.1 Introduction

Quantification is a recently proposed Data Mining task whose objective is to estimate the class distribution in a test set. This task has found numerous applications where the primary interest is *not* in the classification of individuals, but rather the understanding of the behavior

of groups. Typical uses of quantification are, for instance, sentiment analysis when we want to estimate the proportion of positive tweets about a particular product (GONZÁLEZ *et al.*, 2017b); or, mosquito surveillance when we need to calculate the number of disease-carrying mosquitoes captured by an insect trap (REIS *et al.*, 2018).

Quantification and classification are closely related areas. The simplest quantification method, classify and count (CC), merely uses the outcome of a classifier to count how many objects belong to each class. Nevertheless, classification and quantification have fundamental differences, the main one being that classifiers provide predictions for individual data points, whereas quantifiers issue predictions for sets with several instances.

Therefore, we should consider such differences when designing and evaluating quantification methods. In particular, the size of the test set is a characteristic that can affect the performance of quantification methods. In classification, the relationship between the training set size and error rate is a recurring theme of investigation of practitioners and theorists, being the focal point of the statistical learning theory (VAPNIK, 2013). As a consequence, quantification methods that adopt classifiers in intermediate steps naturally inherits this relationship. Therefore, they can benefit from the same theoretical and practical findings in literature. Conversely, the size of the test set is usually not a point of concern for classification. However, the size of the test set influences the quality of a quantifier and must be regarded in the design, analysis and evaluation of new quantification methods.

Thus far, the quantification community has mostly ignored the importance of test set sizes. Unfortunately, such disregard has detrimental effects on the quality of the research in the area. First, it implicitly assumes that a method that performs well for large test sets will perform equally well for small test sets. Such an assumption can be invalidated by a simple experiment as the one shown in Figure 29. In this figure, we summarize a comparison between five well-known quantification methods on several benchmark datasets with test set sizes of 500 instances (A) and ten instances (B). The ranking of the algorithms has significantly changed depending on the size of the test set, even though all other variables remained unchanged.

Second, not explicitly acknowledging the effects of varying test sample size increases the chances of cherry picking in quantification research. For instance, a paper that proposes a new quantification method that relies on estimating several statistics from the test set may deliberately choose to evaluate the proposal only on large test sets, where the method benefits the most: with more data, there is a higher chance of accurately estimating the required statistics, making the method more competitive.

Finally, by ignoring the importance of the test sample size, we disregard this variable in the design of quantification algorithms. As different applications may pose distinct requirements, the community should be focusing on proposing methods that perform well on a variety of test sets sizes. From our experiments, we observe that several approaches perform well for larger test sets with more than 100 instances. On the other hand, for tiny test sets of 10 data points, only two

Figure 29 – Visualization of the Nemenyi’s post hoc test for mean absolute quantification error. Groups of methods which are not significantly different at $p < 0.05$ are connected.

Source: Elaborated by the author.

strategies (Median Sweep and SORD) outperformed the arguably naive and frequently criticized Classify and Count algorithm. In our opinion, this is evidence that the community has given little attention to this setting, and further research needs to be developed to propose accurate methods for small test sets.

Although the smallest test sets are the most challenging to provide accurate results for, we do not think the community should entirely focus on them. We believe that most applications require quantifying sets of varying sizes. For instance, for the sentiment analysis and insect counting applications mentioned before, an important outcome is the number of positive tweets/captured insects over different periods of time, such as the last hour, day, and week. This setting leads to the operation of quantification methods on different test set sizes. We explore this idea by proposing a simple metalearning strategy that selects the quantification algorithm according to the size of the test set.

Our objective, with this chapter, is to call the community attention to the importance of the test set size when designing and evaluating quantification methods. To support our claims, we define a new experimental design to evaluate quantification algorithms. To the best of our knowledge, we perform the most comprehensible evaluation of quantification methods in literature. We compare 12 quantification methods representative of different counting paradigms on 13 benchmark datasets, with 14 test set sizes and 100 class distributions.

The main contributions of this chapter are: (i) we empirically demonstrate the importance of the test set size to assess the performance of quantification methods; (ii) we show that current quantification methods generally have a mediocre performance on the smallest test sets; and (iii) we propose a metalearning scheme to select the best quantifier based on the test set size that can outperform the best single quantification method.

At last, we advocate that, for now on, quantification research should use the experimental setup proposed in this chapter to evaluate their proposals and compare to the state-of-the-art. To support these comparisons, we created a chapter website¹ with all data, code and detailed results so that other researchers can directly use our results when evaluating newly-proposed methods.

This chapter is organized as follows: Section 5.2 provides the necessary background information on the quantification task; Section 5.3 summarizes the relevant literature; Section 5.4 describes the experimental setup proposed in this chapter; Section 5.5 presents our metalearning framework for recommending quantifiers according to each test set; Section 5.6 discusses the experimental setup and Section 5.7 presents the empirical results and discussion of a comprehensible evaluation of quantification methods; Finally, Section 5.8 concludes this chapter and present directions for future work.

5.2 Background

Quantification is a supervised machine learning task that shares several similarities with classification, such as the attribute-value representation for observations that correlate with a nominal class attribute.

Consider the labeled set $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)\}$, in which each example $\vec{x}_i \in \mathcal{X}$ is a vector in the m -dimensional feature space $\mathcal{X}$, and $y_i \in \mathcal{C} = \{c_1, \dots, c_k\}$ is its respective class label. We can formally define a quantifier as a model that predicts the prevalence of each class in a sample according to the following equation:

$$\mathcal{Q} : \mathbb{S}^{\mathcal{X}} \longrightarrow [0, 1]^k$$

where $\mathbb{S}^{\mathcal{X}}$ denotes the universe of possible samples from $\mathcal{X}$. Given a specific sample $S \in \mathbb{S}^{\mathcal{X}}$, the quantifier outputs a vector, $\hat{q}(S) = [\hat{p}_1(S), \dots, \hat{p}_k(S)]$, that estimates the prior probability for each respective class $c_1, \dots, c_k$, subject to the constraint $\sum_{i=1}^k \hat{p}_i(S) = 1$. A good quantifier is one that produces a $\hat{q}(S)$ that reasonably approximates the true class ratios $\vec{q} = [p_1(S), \dots, p_k(S)]$ of the probability distribution from which S was sampled.

All interesting quantification problems undergo a class distribution drift from training to test data. Otherwise, we could use the class ratios from the training set to predict the distribution of the test sets. For our previous examples, sentiment analysis and insect counting, the class distribution in the training set is not a suitable predictor of the class ratio in newly observed data. More importantly, we expect the class distribution to change on multiple occasions over time. On sentiment analysis, the ratio of positive comments can vary with the development of events that affect public opinion or even due to the work of automated trolls commanded by a bad actor. In the insect quantification problem, both relative and absolute numbers of captured insects for

¹ <https://quantification.shinyapps.io/mlqframework/>

each species vary according to the local population, circadian rhythm and ambient factors such as temperature.

We note that several quantification methods for binary quantification rely on classifiers that produce scores. Binary quantification is the task of estimating the positive class ratio in a problem that has only two classes: positive and negative. In this context, the score of one observation is a value that a classification model may produce and that correlates with the probability of said observation belonging to the positive class.

Contrary to classification, which aims to predict each observation's class, quantification focuses on the overall quantity of elements that belong to each class. This distinction implies substantial differences involving experimental design and evaluation metrics. As we will detail later, we are interested in redefining the experimental setup to evaluate new and existing methods.

5.3 Related work

In the last decade, several algorithms for quantification have been proposed. Although they share the same general setting and objective, their introduction by different communities led to different names for the quantification task: *prevalence estimation* (BARRANQUERO *et al.*, 2013), *class prior estimation* (CHAN; NG, 2006), and *class distribution estimation* (GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013).

González *et al.* (2017a) organized most of the prior work according to similarities between algorithms, resulting in a taxonomy of quantification methods described by three different groups:

- I. ***Classify, Count, and Correct***: this group describes all methods that classify each instance firstly and then count how many belong to each class. This group includes the methods that apply correction mechanisms to the result obtained with classify and count;
- II. ***Adapting traditional classification algorithms***: this group contains algorithms that are a modified version of traditional machine learning methods. These methods typically replace the performance metric that the original algorithm aims to optimize with a quantification specific metric;
- III. ***Distribution matching***: the general approach of methods in this group consists in creating a parametric model of the distribution of the training data, and then finding the parameters that best match the distributions of the test data. Several strategies can be applied in order to model the distribution of the data. One frequent technique for binary quantification consists in using histograms to discretely model the density function of scores generated by a classifier, for each class. The weighted averaged of such histograms should then be

matched with the histogram of scores obtained with the unlabeled test set. The weights correlate with the classes ratios.

For our empirical evaluation, we selected quantifiers from all taxonomic groups to provide a significant degree of diversity to our analyses. We describe each one of them in the following section.

5.3.1 Quantification methods

In this section, we briefly introduce several of the most used quantifiers found in literature. For a given test sample T_e , any quantifier algorithm $\mathcal{A}$ produces a quantification vector $q^{\mathcal{A}}(T_e)$ as follows:

$$q^{\mathcal{A}}(T_e) = [p_1^{\mathcal{A}}(T_e), \dots, p_k^{\mathcal{A}}(T_e)]$$

To simplify the contents of this section, we will only describe the estimation of $p_i^{\mathcal{A}}(T_e)$ for each algorithm $\mathcal{A}$ described. Additionally, whenever we describe a binary quantification problem, c_1 corresponds to the positive class and c_0 corresponds to the negative class.

5.3.1.1 Classify and Count (CC)

The arguably simplest quantification method is known as Classify and Count (CC) (FORMAN, 2005). It is the result of a straightforward use of a classifier to count: we induce a classifier h from a training sample T_r with a standard learning algorithm. Then, given a test sample T_e whose class distribution must be estimated, we classify each instance in T_e and estimate the proportion of the classes by counting the classifications and dividing by the cardinality of T_e . Formally, CC is defined as:

$$p_i^{CC}(T_e) = \frac{|\vec{x}_j \in T_e | h(\vec{x}_j) = c_i|}{|T_e|}$$

where $h(\vec{x})$ is the class label assigned to $\vec{x}$ by the classifier h and $|S|$ is the cardinality of S , *i.e.*, the number of elements in S .

Since CC is one of the simplest and most widely used quantification methods, it is worthwhile to briefly analyze its performance. While it is evident that a perfect classifier provides a flawless quantifier, let us examine the more common case in which some error is involved.

Figure 30 illustrates the distribution of classification scores for a binary classification problem with a fixed threshold. Every example with a score above the threshold is classified as positive, otherwise as negative. As we have some overlap between the positive and negative density functions of scores, the classifier is not perfect and will incur some false positive and false negative classifications.

Figure 30 – Threshold and score distribution for a binary classification process (REIS *et al.*, 2018).

Source: Reis *et al.* (2018).

The outcome of CC is the count of examples above the threshold, *i.e.*, the true positives plus the false positives. Even though this particular classifier is not perfect no matter where the threshold is placed, it can still lead to a perfect quantification. To achieve flawless quantification, we need to choose a threshold that results in the same number for both false positives and false negatives. Therefore, when CC counts the number of the true positives plus false positives, such amount will be equal to true positives plus false negatives, which is the correct answer to the quantification problem. Not by chance, that is the case illustrated in Figure 30.

Unfortunately, these are all the good news for CC. As discussed in Section 5.2, interesting quantification problems will present some form of class drift in the test set. In the previous example, if the number of positive observations increases, as illustrated in Figure 31, CC will underestimate the positive class count, since the number of false negatives will also increase while the number of false positives will remain the same. Similarly, this method will overestimate the positive class ratio if the number of positive observations decreases, presenting a systemic error.

Figure 31 – An increase in the number of cases of the positive class makes CC underestimates this class.

Source: Elaborated by Denis dos Reis.

At this point, it should be clear why setting the threshold for perfect quantification is difficult. Such a decision would require accurately estimating the false positive and negative rates, which depend on the class distribution that is the information we ultimately want to discover.

Additionally, for any imperfect classifier, it is easy to prove that the error of CC increases linearly as a function of the true positive ratio whenever it is lower or greater than the one that results in perfect quantification. For the interested reader, (TASCHE, 2016) proceeds with a more in-depth analysis of the limits of CC and quantification without adjustments, in general.

For those reasons, several researchers do not recommend the use of CC for quantification tasks (TASCHE, 2016; REIS *et al.*, 2018). Further, CC is regarded as a baseline that must be outperformed by more refined methods.

5.3.1.2 Adjust Classify Count (ACC)

It is an extension of the CC approach that applies a correction to the counts produced by CC. Intuitively, in the binary quantification case, we can fix CC by subtracting false positives and adding false negatives (*cf.* Figures 30 and 31). Such a correction would result in a perfect classifier independently of the classifier accuracy.

Although less intuitive, a common form to express ACC to predict the positive class ratio in a binary quantification problem is in terms of the true positive (*tpr*) and negative (*fpr*) rates (FORMAN, 2005):

$$p_1^{ACC}(T_e) = \frac{p_1^{CC}(T_e) - fpr_{T_r}}{tpr_{T_r} - fpr_{T_r}}$$

ACC is proven to provide a perfect quantification when the true *tpr* and *fpr* are known. However, calculating the true *tpr* and *fpr* would require a labeled test set. In practice, researchers estimate such quantities using the training set and some error estimation sampling, such as cross-validation.

ACC can also be used for multiclass quantification. In this case, estimates for the probabilities $P(c_i|c_j)$ are needed, where $P(c_i|c_j)$ is the probability of an example that belongs to class c_j being classified as c_i . Such information can be estimated the same way as *fpr* and *tpr*, including the same caveats.

5.3.1.3 Probabilistic Classify and Count (PCC)

PCC is a probabilistic variant of Classify and Count (CC) described by Gao and Sebastiani (2016). The main idea is that probabilities may have richer information than the discrete decision. Like CC, a classifier is learned from a training set T_r and used to estimate the probability of each instance in a test set T_e belonging to each class in $\mathcal{C}$. Then, the prevalence each class is estimated as the average of the probabilities predicted for the said class:

$$p_i^{PCC}(T_e) = \frac{1}{|T_e|} \sum_{\vec{x}_j \in T_e} \hat{p}(c_i | \vec{x}_j)$$

where $\hat{p}(c|\vec{x})$ is the estimated probability of instance x belonging to c , according to the classifier trained. Like CC, PCC will underestimate or overestimate the actual prevalence when the distribution of the classes changes (TASCHE, 2014).

5.3.1.4 Probabilistic Adjust Classify and Count (PACC)

This method is a probabilistic variant of ACC method, that adjusts the estimate of PCC. Proposed by Bella *et al.* (2010) and named Scaled Probability Average (SPA). In the binary case:

$$p_1^{PACC}(T_e) = \frac{p_1^{PCC}(T_e) - fpr_{T_r}}{tpr_{T_r} - fpr_{T_r}}$$

5.3.1.5 Threshold selection methods

The performance of ACC deteriorates if the class distribution in the training set is highly imbalanced. In the binary case, it happens, for instance, when a classifier is learned using a training set where positive examples are scarce. In that scenario, the classifier tends to predict the majority class, *i.e.*, the negative class and consequently reduces the number of false positives. However, this results in a low tpr , leading to a low denominator in Equation 5.3.1.2 and incurring substantial corrections. To overcome this issue, Forman (2006) proposes several approaches to select a threshold value for the classifier coupled with ACC:

X: selects the threshold where the curve of false positive rate and false negative rate ($1 - tpr$) crosses each other, *i.e.*, where $fpr = 1 - tpr$. This method avoids the tails of both curves;

Max: searches for the largest denominator which is obtained by selecting the threshold that maximizes $tpr - fpr$;

T50 selects the threshold where $tpr = 50\%$. This method also avoids the tail of tpr curve;

Median Sweep (MS): method estimates true positive rate tpr and false positive rate fpr for a range of thresholds through cross-validation on the training set and then estimates the positive prevalence in the test set for all thresholds. After that, it returns the mean or median of the estimated positive prevalence.

5.3.1.6 Expectation Maximization Quantification (EMQ)

Saerens, Latinne and Decaestecker (2002) proposed an iterative approach to estimate the class prevalence in imbalanced class distribution scenarios using the Expectation Maximization

algorithm. Primarily, the class prevalence of each class is initialized to be the class prevalence in the training set. Then, with each iteration, estimates of the class ratios updated to better approximate the distribution of T_e . Formally:

$$\begin{aligned}\hat{p}_i^{EMQ\ at\ (0)}(T_e) &= \hat{p}_i(T_r) \\ \hat{p}^{(t)}(c_i|\vec{x}_l) &= \frac{\frac{\hat{p}_i^{EMQ\ at\ (t)}(T_e)}{\hat{p}_{T_r}(c_i)} \hat{p}_{T_r}(c_i|\vec{x}_l)}{\sum_{j=1}^k \frac{\hat{p}_j^{EMQ\ at\ (t)}(T_e)}{\hat{p}_j(T_r)} \hat{p}_{T_r}(c_j|\vec{x}_l)} \\ \hat{p}_i^{EMQ\ at\ (t+1)}(T_e) &= \frac{1}{n} \sum_{l=1}^n \hat{p}^{(t)}(c_i|\vec{x}_l)\end{aligned}$$

where $\hat{p}_{T_r}(c_i|\vec{x}_l)$ and $\hat{p}_i(T_r)$ are the estimated probability of instance $\vec{x}_l$ belong to class c_i and the proportion of class c_i , both according to the training set. For each iteration t , $\hat{p}^{(t)}(c_i|\vec{x}_l)$ and $\hat{p}^{EMQ\ at\ (t)}(c_i)$ are re-estimated sequentially for each instance $\vec{x}_l \in T_e$ and each class $c_i \in \mathcal{C}$. In our experiments, we defined five iterations as a stopping condition.

5.3.1.7 Quantification Trees (QT)

[Milli et al. \(2013b\)](#) introduce a decision tree learning algorithm for quantification called Quantification Trees (QT). Similar to other decision trees, their proposed algorithm also greedily selects the splitting-feature and splitting-threshold at a decision node that minimizes or maximizes some criterion. However, instead of using classic measures, the proposal applies a quantification measure for selecting the best split. The evaluation metrics used in their paper are briefly described below:

Classification Error Balancing: For each possible split and class c_i , the difference between false negative (FN_i) and false positive (FP_i) is computed, as:

$$Ec_i = E_i = |FP_i - FN_i|$$

where the optimum value for quantification is achieved when the value of FP_i is equal to FN_i , i.e., $E_i = 0$;

Classification-Quantification Balancing: this approach is an improvement over the previous one. The authors made a trade-off between quantification and classification accuracies. Thus, for each possible split and class c_i , the following equation is applied:

$$\overline{E}c_i = |FP_i - FN_i| \times |FP_i + FN_i|$$

where the right side of the multiplication is a measure for classification performance, perfect classification is achieved if $FP_i = FN_i = 0$.

For a possible split, the values for the chosen evaluation metric are aggregated in a new vector: $\vec{Q} = [Ec_1, \dots, Ec_k]$ or $\vec{Q} = [\bar{E}c_1, \dots, \bar{E}c_k]$. The final score is given by the L2-norm of $\vec{Q}$. To calculate the goodness of the split, the quantification accuracy of parent node (before splitting) and child node (after splitting) are compared:

$$\Delta = ||\vec{Q}^{\text{parent}}||_2 - ||\vec{Q}^{\text{child}}||_2$$

where Δ denotes the goodness of the split: the larger the delta the better the split. The growing process is terminated if $\Delta \leq 0$. Additionally, the authors performed experiments with a simple decision tree and with Random Forest. In our experiments, we used only Random Forest with Classification Error Balancing to define the splits. After the classification model is induced, class distribution is estimated by applying CC with said model.

5.3.1.8 HDy algorithm

Proposed by [González-Castro, Alaiz-Rodríguez and Alegre \(2013\)](#). Consider a classifier induced with T_r and that is capable of issuing scores. HDy uses two normalized histograms, P^+ and P^- , that summarize the samples S^+ and S^- , respectively. S^+ is composed of scores generated by the classifier for positive examples in a labeled validation set. S^+ is the analogous for negative examples. Given a test set, a histogram $Q(T_e)$ summarizes the set of scores $S(T_e)$, which is obtained using the same classifier on the test set T_e . These histograms, namely P^+ , P^- , and $Q(T_e)$, represent the distributions of the training set for each class, and the distribution of the test sample, respectively. Finally, the positive class ratio is estimated as follows:

$$p_1^{\text{HDy}}(T_e) = \arg \min_{0 \leq \alpha \leq 1} \{ \text{HD}(\alpha P^+ + (1 - \alpha) P^-, Q(T_e)) \}$$

where HD is the Hellinger distance, and b is the number of bins in the histograms. The positive class ratio is found by linear search where through several possible values of α . In ([GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013](#)), the authors estimate the positive ratio for many number of bins: from 10 to 100 with increments of 10. The final positive class ratio reported is the median of the ratios estimated.

[Reis et al. \(2018\)](#) proposes to use Ternary Search to make HDy more efficient and generally more precise than through linear search. They also include a new variation of HDy with Laplace smoothing (HDy-LP), making this more reliable on extreme scenarios.

5.3.1.9 DyS framework

Maletzke *et al.* (2019) argue that HDy is an instance of a more general framework. They propose a framework named DyS. According to this framework, the positive class ratio is estimated as follows:

$$p_1^{\text{DyS}}(T_e) = \arg \min_{0 \leq \alpha \leq 1} \{ \text{DS}(\alpha H(S^+) + (1 - \alpha)H(S^-), H(S(T_e))) \}$$

where DS is a dissimilarity function, S^+ , S^- , $S(T_e)$ are the sample of scores for the positive training data, negative training data, and test sample, respectively, and H is a function that converts a sample of scores into a representation that is compatible with DS and that supports mixing two distributions according to a factor α .

The authors conducted an experimental evaluation with distinct similarity functions and have provided a rank of better functions. They also redefined the number of histogram bins used to model the score distribution. Based on that, in our experiments, we selected the top five similarity functions: ORD, Sample ORD (SORD), Probabilistic Symmetric (PS), Jensen Difference (JD), and Topsøe (TS). In this chapter, we report the positive class ratio that is the median for a varying number of bins, from 2 to 20 with increments of 2, following the findings of (MALETZKE *et al.*, 2019).

5.3.2 How test set sizes affect the performance of the quantification methods

At this point, we are in the position to review each of the surveyed methods and discuss how the test set size may affect their counting performance. To facilitate our analysis, we categorize them into three classes:

Methods that make no use of test set data: In this category falls all methods that rely only on training data to induce classifiers and extract the necessary statistics to fix the quantification outcomes. The most straightforward approaches are CC and PCC, but this category also includes ACC and its variations such as PACC and the threshold selection methods proposed by Forman (2006). As the methods in this category do not rely on test set information, they will not be affected by the test set size. Conversely, to compute unbiased statistics or make impartial threshold selections, ACC variations depend on sampling methods for error estimation. This dependency may incur in the need of a reasonably-sized labeled set that is a representative sample of all classes.

Methods that make incremental use of test data: EMQ is a method that incrementally uses the test data to adjust a count estimate initially set as the training class ratio. The intuition of this method is that, for a suitable-sized test set, the EMQ approach will have enough

iterations to converge to an accurate test set quantification. However, for small test sets, there may not have enough data, and the training set ratio may drive the final output. EMQ attempts to overcome this limitation by allowing multiple interactions over the test set.

Methods that use the whole test sample: This final category includes methods such as HDy and DyS. These methods classify the entire test set sample to estimate the distribution of the scores and use a histogram with b bins approximates such a distribution. It is clear that, with a lack of test data, the approximation of score distribution will be poor, impacting the accuracy of the methods. [Maletzke et al. \(2019\)](#) discusses this issue and makes a comprehensive analysis of the impact of the parameter b in the performance of these methods. In summary, smaller values of b are more suitable for small test set sizes, since they avoid having many zero-valued bins. Particularly, DyS with the distance functions ORD and SORD is less sensible to this effect, since ORD takes the relation of different bins into consideration, and SORD operates directly over observations instead of a histogram. Nonetheless, ORD and SORD can still be negatively affected by a smaller test sample, since there is fewer observations to accurately approximate the true distribution that is later compared.

For these reasons, we expect to find out a significant impact of the test set size, which can be more prominent over the last described class of quantifiers.

5.3.3 Experimental design

Experiments in quantification are usually more involved than those in classification. The evaluation process is data hungry since it requires measuring the performance on entire samples with several data points. Moreover, we need to repeat the experiment to obtain confidence intervals, collect the test samples from a larger pool to guarantee some diversity among the samples, and vary the class distribution to simulate the expected data distribution variability.

Due to the similarity between the quantification and classification tasks, quantification experiments are typically conducted using classification datasets. The experimental setup usually follows these two steps:

- Each dataset is divided into two subsets using stratified sampling, resulting in training and test sets;
- Given the test set of size n_{Te} examples, a collection of samples of length $m_{Te} \ll n_{Te}$ is extracted varying the class distribution randomly or according to a predefined range.

This approach allows simulating a range of class distribution drifts, and has been used by several authors, including [Forman \(2005\)](#), [Bella et al. \(2010\)](#), [Barranquero et al. \(2013\)](#), [Milli et al. \(2013b\)](#), [Reis et al. \(2018\)](#) and [Pérez-Gállego et al. \(2019\)](#), [Maletzke et al. \(2019\)](#).

However, to the best of our knowledge, every paper in quantification research uses a fixed and arbitrary m_{Te} . The only exception is the paper of [Maletzke et al. \(2019\)](#). However, in that paper, the authors are interested in characterizing the performance of a smaller set of quantification methods, mixture models, and understand the importance of their internal parameters.

5.4 Designing experiments in quantification

The experimental design used in the vast majority of quantification papers ignore the influence of the test set size. In this chapter, we argue this is a severe flaw since the performance of quantifiers fluctuates according to this variable. Identifying the performance in a single test sample size does not provide enough information to characterize the behavior of a method. For this reason, we include an additional step into the standard evaluation procedure so that we can assess the performance on distinct test sizes, in addition to varied class distributions. Algorithm 5 describes the steps to generate samples for the evaluation.

Algorithm 5 – Generating test samples

Input: Test set T_e

Output: Set of test samples S_{T_e}

```

1:  $S_s \leftarrow \{s_{min}, \dots, \min(L, |T_e^{c_1}|, \dots, |T_e^{c_k}|)\}$     ▷ Sample sizes,  $s_{min} > 1$  and  $L$  is a hard limit
2:  $Pr_{class} \leftarrow \{[p_1^1, \dots, p_k^1], \dots, [p_1^N, \dots, p_k^N]\}$     ▷  $N$  class distributions
3:  $R_{sample} \leftarrow R$     ▷ Replication
4: for  $i \leftarrow 1, \text{length}(S_s)$  do
5:   for  $j \leftarrow 1, \text{length}(Pr_{class})$  do
6:     for  $k \leftarrow 1, R_{sample}$  do
7:        $S_{T_e} \leftarrow S_{T_e} \cup \{getSample(T_e, S_s[i], Pr_{class}[j])\}$ 
8:     end for
9:   end for
10: end for
    return  $S_{T_e}$ 

```

In Algorithm 5, $T_e^{c_i} \subset T_e$ includes all observations from T_e that belong to class c_i . The function *getSample* returns a sample, given the number of observations and class distribution, using the examples from the T_e . For most datasets, due to limited data, different samples can share observations. However, one observation should not appear more than once in a same sample.

5.5 Metalearning quantification

We argue that the best quantification algorithm for large sets may not be the best method for small sets, and vice versa. We later empirically confirm this hypothesis. Thus, two strategies could be applied to improve our evaluation procedure. First, the proposal of new quantifiers that

can perform well independently of the size of the test set. Second, a mechanism to autonomously select the quantifier, from a group of options, that is the most adequate for a given test set. In this chapter, we adopt the latter strategy: our proposal is not a new quantification algorithm by itself, but rather a framework that recommends quantifiers according to each test set. To achieve this effect, we employ an algorithm recommendation method with metalearning. As stated by [Brazdil et al. \(2009\)](#), metalearning systems are used to indicate which algorithm should be used to reach the best possible results for each individual task, according to its particularities.

Our framework, named Metalearning Quantification (MLQ), applies metalearning concepts to describe each problem and learn a recommender that predicts which quantifier should be used for each test set. Figure 32 shows the architecture of our framework.

Figure 32 – Framework architecture for quantifiers recommendation.

Source: Elaborated by the author.

First, we gather several datasets and split each one into two halves (training and test halves). The training half is used to learn all quantifiers that can be recommended for quantifying sets, which are extracted from the testing half. We also use the training half to learn our recommender, following path two in Figure 32. Then, we break up the training half into two new halves, which are the training and validation sets for the metalearning procedure. From the

validation half, we extract several test sets of different sizes and class distributions. For each one, we extract metafeatures and measure the performance of all quantifiers. As a result, a tuple composed of metafeatures and quantifier performance is recorded in a metatable (item four in Figure 32). We repeat this process several times to get a vast diversity of tuples. After that, this metatable is used to induce our recommendation model, which is a classifier where the classes represent different quantification algorithms.

Metalearning literature reports several metafeatures to describe characteristics of a dataset or a group of data points, such as information theory and statistics measures (LEMKE; BUDKA; GABRYS, 2015). In our framework, in addition to the test size itself, as a numerical feature, we use simple statistic measures and the distance reported by the DyS framework when quantifying (MALETZKE *et al.*, 2019). The statistics measures used were: coefficient of variation, kurtosis, skewness, and correlation. The average value of all attributes was calculated to describe a set of observation points.

Finally, we follow the path three in Figure 32 to build the final quantifiers that can be chosen by the recommender during our test phase, where we evaluate the methods. Path five represents the experimental evaluation, described in the next section.

5.6 Experimental setup

In this section, we describe a set of experiments with several well-known quantification algorithms, aiming to show the influence of varying test set size on their performance. Table 10 presents all binary quantifiers evaluated in our experiments.

DyS framework was evaluated including the top five similarity functions according to Maletzke *et al.* (2019): ORD, Sample ORD (SORD), Probabilistic Symmetric (PS), Jensen Difference (JD), and Topsøe (TS). We report the positive class ratio that is the median for a varying number of bins, from 2 to 20 with increments of 2, following the findings.

We must note that, in our experiments, we only used binary quantification datasets. This allowed us to test a wider range of algorithms and simplify the design of experimental evaluation. However, limiting our setup to binary quantification does not disqualify the point we are making regarding the relevance of test sample size. Indeed, if this variable affects binary quantification, it stands to reason that it also affects multiclass quantification.

We uniformly split each dataset into two parts: training and test. With the training part, we performed 10-fold cross-validation to obtain the classification scores used by the mixture models that follow the DyS framework, as well as to calculate the *tpr* and *fpr* performance statistics used by ACC and its variations. The full training half was also used to train a single scorer that was applied on the individual data points in the test half, to obtain a set of test scores. We produced all scores using Random Forests with 200 trees. For PCC and PACC methods we

Table 10 – Quantifiers evaluated.

Quantifier	Acronym	Taxonomy Group
Classify and Count	CC	I
Adjust Classify and Count	ACC	
Probabilistic Classify	PCC	
Probabilistic Adjust Classify and Count	PACC	
Threshold Selection Method		
Set the decision threshold where $tpr = 50\%$	T50	
Set the decision threshold where $(1 - tpr) = fpr$	X	
Set the decision threshold where $tpr - fpr$ is maximized	MAX	
Estimate tpr and fpr for several thresholds, returning the median of them	MS	
Quantification Trees	QT	II
Expectation Maximization Quantification	EMQ	III
Distribution matching with Hellinger Distance	HDy	
Mixture Model Framework	DyS	

Source: Elaborated by the author.

calibrate random forest scores, using an isotonic regression calibration based on pair-adjacent violators (NAEINI; COOPER, 2016). Test samples were obtained from the test part according to Algorithm 5. We also use Random Forests to induce the recommendation model for MLQ framework.

We varied the test set size from 10 to 100 with increments of 10 data points, and from 100 to 500 with increases of 100. For each test set size, we varied the positive class proportion from 0% to 100% with increments of 1%. We execute 10 runs for each pair of test set size and class distribution.

As the time complexity of SORD grows linearithmic with the total number of observations involved, we undersampled training scores to 1,000 per class, whenever there were more than that. For the EMQ approach, we used five iterations as a stopping condition.

We measured the performance of quantifiers in our experiments using the Mean Absolute Error (MAE) (SEBASTIANI, 2018). MAE is the average of absolute differences between true ($\vec{q}$) and predicted ($\hat{\vec{q}}$) quantification for a set of classes $\mathcal{C}$. The performances were compared using the Friedman test with 95% confidence level and Nemenyi post hoc test.

Table 11 presents a brief description of the datasets used in our experiments obtained from UCI (DHEERU; TANISKIDOU, 2017), OpenML (VANSCHOREN *et al.*, 2013), and Reis (REIS *et al.*, 2018) repositories. Specific citations are requested for HTRU2 (LYON *et al.*, 2016), Mozilla4 (KORU; ZHANG; LIU, 2007), Nomao (CANDILLIER; LEMAIRE, 2012), and Occupancy Detection (CANDANEDO; FELDHEIM, 2016).

Table 11 – Datasets description.

ID	Dataset	Size	Features	Repository
A	AedesSex	24,000	27	Reis
B	Anuran Calls	6,585	22	UCI
C	ArabicDigit	8,800	27	UCI
D	BNG (vote)	39,366	9	OpenML
E	Spambase	4,601	57	UCI
F	Handwritten-QG	4,014	63	Reis
G	Wine Type	6,497	12	UCI
H	Wine Quality	6,497	12	UCI
I	HTRU2	17,898	8	UCI
J	Letter Recognition	20,000	16	UCI
K	Mozilla4	15,545	5	OpenML
L	Nomao	34,465	118	OpenML
M	Occupancy Detection	20,560	5	UCI

Source: Research data.

A couple of observations about the datasets are due. First, Wine Type dataset is similar to Wine Quality. However, we want to differentiate between white and red wines, rather than the wine quality. Second, ArabicDigit is a preprocessed version of the original so that all examples have the same number of features (REIS *et al.*, 2018), and the objective is to predict the sex of the speaker rather than which digit is spoken.

5.7 Experimental evaluation

We open this section by showing the quantification results for all algorithms and datasets, considering only our largest test set size of 500 observation points. Figure 33 shows the critical difference diagram.

Figure 33 – Visualization of the Nemenyi's post hoc test for mean absolute quantification error. Test sample composed of 500 cases. Groups of methods which are not significantly different at $p < 0.05$ are connected.

Source: Elaborated by the author.

According to the results, algorithms based on distribution matching achieve the best

Figure 34 – Mean absolute error variability for test set size and class distribution variability of two instances of the framework DyS.

Source: Elaborated by the author.

performance. Particularly, four instances of the DyS framework (MALETZKE *et al.*, 2019) are among the top five quantifiers. These results are not surprising, given the nature of these quantifiers, which are benefited by test sets with a large number of instances, allowing them to model the data distribution precisely.

These results are a typical summary of the experimental results we can find in quantification literature. In short, a set of algorithms compared over a collection of datasets for an arbitrary test set size. However, as discussed in Section 5.1, such results would only be representative of the whole spectrum of test set sizes if the performance of the quantification algorithms were independent of such a variable.

To illustrate that this assumption does not hold, Figure 34 shows the mean absolute error for all test set sizes and class distributions, for each one of the two best-ranked algorithms according to the previous results in Figure 33. Due to lack of space, our supplemental material website contains figures for all other algorithms².

Figure 34 exhibits small errors across all class distributions for test set sizes with more than 100 instances. However, for smaller test sets, with less than 100 cases, the errors tend to increase quickly. A worrisome aspect of these results is the fact that the literature on quantification has mostly ignored small test sets in their evaluations.

Motivated by our initial results, we reapply the same statistical analysis for all algorithms and a tiny test set size of ten observations. Figure 35 shows the critical difference diagram for all quantification approaches. The rank of the algorithms has significantly changed compared to the ranking built with test sets of 500 observations.

For large test set sizes, instances of the framework DyS produced the best results followed by MAX and X, which are algorithms from threshold selection family. We also note that HDy-LP

² <<https://quantification.shinyapps.io/mlqframework/>>

Figure 35 – Visualization of the Nemenyi’s post hoc test for mean absolute quantification error. Test sample composed of 10 cases. Groups of methods which are not significantly different at $p < 0.05$ are connected.

Source: Elaborated by the author.

performed poorly, albeit it being similar to DyS. This behavior was reported by [Maletzke et al. \(2019\)](#) as well, where the authors noted the influence of the number of bins used to describe score distributions.

Conversely, MS was the best-ranked algorithm for the smallest test set size followed by SORD (which is also one instance of DyS), and CC. In the literature, MS is reported as one of the best quantification algorithms while the CC is considered the naive strategy and frequently regarded as a baseline. We also note that EMQ performs better in small test sets than in large ones. On the other hand, PCC, PACC, and QT perform poorly in all scenarios (large and small test set sizes). For QT algorithm, we used source code provided by the authors ([MILLI et al., 2013b](#)).

Moreover, SORD and DyS+ORD were the only instances of DyS with no statistical difference against MS when the test set size is composed of ten cases. The explanation is that ORD is less affected by sparse histograms than other distance functions, since it can compare misaligned histogram bins. SORD does not use histograms at all.

Another interesting finding is that there is statistical difference between DyS+PS and MS in both scenarios, even though DyS+PS was the best performing algorithm in one setting, and MS was the best in the other one. Nevertheless, we must keep in mind that these results were obtained for several datasets from different domains. Each dataset can be analyzed separately. For the sake of clarity, we show in Figure 36-left the mean absolute error obtained by both DyS+PS (which is the top-ranked for the largest test sets) and MS (which is the top-ranked for the smallest test sets) for all datasets.

By analyzing the errors in Figure 36-left, we would probably opt to use DyS+PS rather

Figure 36 – Mean absolute quantification error obtained with DyS+PS and MS. For test sets composed of 500 cases (*left*) and composed of 10 cases (*right*).

Source: Elaborated by the author.

than MS. If we chose DyS+PS as our quantifier for all datasets, we would get the best results for large test sets. However, it is not the best decision if the test set size is expected to change drastically. Figure 36-*right* compares the results of DyS+PS and MS for the same datasets but when the test sets comprise ten observations. For small test sets, MS performs better than DyS+PS for a majority of the datasets, and that difference is statistically significant ($p - \text{value} < 0.05$).

Our results demonstrate that the experimental evaluations conducted until now by the literature have an incomplete analysis, since they neglect a relevant aspect of real-world problems. Consequently, current analyses provide inaccurate recommendations for quantification tasks.

In our subsequent analysis, we consider all quantification algorithms and all test set sizes. Figure 37 shows the ranking of all algorithms. Instances of the framework DyS perform better than other methods, except when it is using ORD as similarity function. As previously said, DyS+ORD, differently from the other instances, is less affected by sparseness, which seems less relevant for large test sets.

Figure 38 presents the critical differences diagram of the Nemenyi's test for each quantification algorithm over all datasets.

The statistical analysis reveals that four of the five DyS instances evaluated are ranked as top quantifiers when we vary the test set size. SORD is the best quantifier, but it only performs better than HDy-LP, PACC, MS, PCC, and QT with a statistical difference ($p - \text{value} < 0.05$).

One important caveat of the previous results is that, as we varied the sample size from 10 to 100 with increments of 10, and from 100 to 500 with increments of 100, the results are biased towards the measurements made on smaller test sets.

At last, we apply our proposed metalearning framework, MLQ, to select one of the

Figure 37 – Aggregation of several rank positions for all quantification algorithms. Test sample size varied from 10 to 500.

Source: Elaborated by the author.

Figure 38 – Critical difference diagram among all algorithms. Test sample size varied from 10 to 500. Groups of methods which are not significantly different at $p < 0.05$ are connected.

Source: Elaborated by the author.

following quantifiers: (i) MS, which is the top-ranked for small test sets (Figure 35); (ii) DyS-PS, the best ranked on large test set size (Figure 33); and (iii) SORD, which has performed better than others when test set size has varied (Figure 37).

Figure 39 shows the ranking of all algorithms when the test set size varies, including the MLQ framework.

Our metalearning framework improves on the best quantifiers and outperform them in a more broad scenario, where test sample size varies. This achievement corroborates with our initial hypothesis. We have argued that the test set size should be considered in the experimental evaluation of quantifiers, guiding practitioners to make the best choice regarding quantification

Figure 39 – Aggregation of several rank positions for all quantification algorithms, including MLQ framework. Test sample size varied from 10 to 500.

Source: Elaborated by the author.

algorithm for each domain.

Although MLQ has decreased quantification errors when test set size varies, it is unclear how helpful was the metalearning scheme. Ultimately, we fed the method with overly simple metafeatures and the best quantifiers for each scenario. To obtain a more conclusive analysis, we investigate the real contribution of our metalearning scheme and how far the MLQ results are from the ideal scenario. To achieve that, we define the *topline* (TOP) and *baseline* (RND) schemes. The *topline* scheme selects the best quantifier for each test set and the *baseline* makes this selection randomly. Figure 40 shows the mean absolute error for MLQ, *topline*, and *baseline* schemes.

As expected the *topline* (TOP) performs better than our MLQ and the *baseline*. However, our proposal outperforms the *baseline*, indicating that our metalearning scheme actually learned how to properly select a quantification model, even though it can potentially be improved to get closer to the *topline*.

5.8 Conclusion

This chapter is the first to question the relevance of test set size in the design and evaluation of quantification methods. We defined a new experimental design to evaluate quantification algorithms in a comprehensive manner. Unfortunately, the literature has mostly ignored the importance of test set sizes, which has impacted on the quality of the research in the area.

Figure 40 – Mean absolute quantification error of MLQ, TOP, and RND. For test set varied from 10 to 500.

Source: Elaborated by the author.

In our experiments, we evaluated, using our experimental design, 12 quantification methods that are representative of different paradigms on 13 benchmark datasets, with 14 test set sizes and 100 class distributions, and we concluded that:

1. The test set size is an important variable to evaluate the performance of quantification methods. The analysis of a newly-proposed algorithm is incomplete without analyzing its performance with different test set sizes;
2. Current sophisticated quantification methods generally have a mediocre performance on tiny test sets, frequently underperforming Classify and Count, a baseline method;
3. A strategy for choosing which quantifier should be adopted depending on the test sample can outperform the best quantification methods when all test set sizes are considered.

We explore those findings by proposing a simple metalearning approach that selects the quantification algorithm according to metafeatures of the test set, including its size. This simple proposal has outperformed all binary quantifiers evaluated, evidencing how relevant is the test set size for the quantification performance. We also show that metalearning can be a promising tool to provide a generic mechanism to quantify samples of data under different scenarios, beyond varying size.

As a part of our future work, we will analyze the impact of the score quality on quantification methods.

ACCURATELY QUANTIFYING UNDER SCORE VARIABILITY

Quantification is a prominent task in Data Mining that has received an increasing amount of attention in the last years. The objective is to predict the class distribution of a data sample. Most of the quantification algorithms are evaluated in well-controlled scenarios that rely on the assumption that the only change from training to test data is in the prior probability of the classes. This chapter is the first to show how binary quantifiers are affected when the quality of the scores outputted by classifiers vary. Such variation can have many sources such as concept drifts and the presence of negative subclasses. We show that well-known methods will not only underperform when the problem becomes harder, as one expects but also when it becomes easier. Thus, we propose a new quantifier algorithm that accurately quantifies while allowing changes in the distribution of scores. We show that our proposal outperforms the state-of-the-art in a comprehensive empirical evaluation involving 11 methods and six datasets.

6.1 Introduction

Quantification is a prominent task in Data Mining that has received an increasing amount of attention in the last years. In this task, the objective is to predict the class distribution of a data sample. Contrary to classification, quantification concentrates on the understanding of the behaviour of groups, rather than the recognition of individuals. Quantification finds applications in diverse domains of study, such as sentiment analysis, where the objective is to estimate the proportion of positive tweets about a particular product ([GONZÁLEZ *et al.*, 2017b](#)). Another example is mosquito surveillance, where the goal consists in calculating the number of *Aedes aegypti* mosquitoes captured by an insect trap ([SILVA *et al.*, 2015a](#)).

Although classification and quantification are two different tasks, quantification frequently avails of classification methods as an intermediate step. For instance, the most straightfor-

ward quantifier, named Classify and Count (CC), consists of classifying each example and then counting how many of them belong to each class. Although simple, CC has flagrant shortcomings, including a systemic error introduced when the class distribution drifts (DIETTERICH; KONG, 1995). A more reliable approach is Adjusted Classify and Count (ACC), which fixes the bias in CC by adopting a correction factor based on the model's true (tpr) and false (fpr) positive rates.

ACC has an outstanding property. It provides perfect quantification results independent of the classifier accuracy, if precise estimates of tpr and fpr are provided. In practice, several factors, such as lack of data and class imbalance, hinder the estimate of such statistics. Therefore, ACC frequently is unable to achieve perfect results.

The symbiosis between classification and quantification goes far beyond CC and ACC. The quantification community has proposed a large number of methods. The majority relies on binary classifiers that output a score representing the confidence in a positive classification as an intermediate step for quantification.

Therefore, it seems natural to question what is the dependency of quantification accuracy of the underlying scorer quality. In one side of the spectrum, as CC implies, a perfect scorer would give rise to a perfect quantifier. On the other side, the development of ACC bias correction suggests a perfect quantifier is feasible independently of the scorer quality.

Recently, Tasche (2019) demonstrated that the score quality has a significant impact over confidence intervals outputted by quantification methods. Hence, empirically establishing a connection between classification and quantification accuracy. In this chapter, we take a step further and investigate the impact that changes in the score quality, from training data to test data, can have in quantification accuracy. We hypothesize that changes in the data distribution can affect the score distribution produced by classifiers and consequently impacting distinct quantification method differently.

Although our research does not assume any particular form of data distribution drift, we discuss a simple and common form of change to illustrate the applicability of our research. In the mosquito surveillance problem, the main task is to count the number of captured individuals from a specific species. For instance, consider the positive class the *Aedes aegypti* female mosquito, the primary vector of several diseases such as Yellow, Dengue, Zika and Chikungunya fevers. Naturally, the negative class represents all other classes that might be captured by the surveillance system.

Therefore, several unknown subclasses compose the negative class, which distributions may vary independently from each other. Given that some subclasses can be more similar to the positive class than others, each batch of captured insects may produce scores with a different overlapping degree.

The first contribution of this chapter is to investigate the impact of the underlying classifier in the quantification performance. We assume the classifiers provide a score. Therefore,

the quantifiers operate over a score distribution derived from instances present in a sample to be quantified. We simulate various degrees of overlap between the positive and negative score distributions indicating problems with different difficulty degrees. More importantly, we analyze the behaviour of state-of-the-art quantifiers trained with a given score distribution as such distribution changes. We show that these methods will not only underperform when the problem becomes harder, as one would expect, but also when it becomes easier.

From this first analysis, it becomes clear that any change in the score distribution relative to the problem difficulty will significantly affect the performance of quantifiers. The second contribution of this chapter is a method that accounts for score distribution changes in the quantification estimates. We augment the recently proposed quantification framework DyS (MALETZKE *et al.*, 2019) to account for score distribution changes. We show the proposal outperforms the state-of-the-art in a comprehensive empirical evaluation involving 11 methods and six datasets.

6.2 Background

Quantification is a supervised machine learning task that shares critical similarities with classification, the main one being the attribute-value representation for observation points and their relation to a nominal class attribute.

Formally, in the case of binary classification, let $D = \{(\vec{x}_1, y(\vec{x}_1)), \dots, (\vec{x}_n, y(\vec{x}_n))\}$ be the labeled set, in which each example $\vec{x}_i \in \mathcal{X}$ is a vector in the m -dimensional feature space $\mathcal{X}$, and $y(\vec{x}_i) \in \mathcal{Y} = \{\oplus, \ominus\}$ is its respective class label. In such a setting, we can define a binary classifier as a model h_c induced from D such that:

$$h_c : \mathcal{X} \longrightarrow \{\oplus, \ominus\}$$

The objective of h_c is to predict, which is the class label of previously unseen examples as accurately as possible. In this scenario, a scorer h_s is often adopted to achieve the proposed goal:

$$h_s : \mathcal{X} \longrightarrow \mathbb{R}$$

which aims to predict, for each example, a numerical value that correlates to $\Pr(\oplus|\vec{x}_i)$, that is, the posterior probability of an example belonging to the positive class. If the scores are within $[0, 1]$ and accurately depict the posterior probability, they are calibrated scores. To simplify the remaining of the text, we refer to scores of positive examples as positive scores, and scores of negative examples as negative scores. We can conveniently convert a scorer into a classifier by setting a threshold: only examples scored above the threshold are classified as positive.

In binary quantification, we are not interested in individual class labels. Instead, we predict the proportion of the examples in a test sample that is positive. A quantification model h_q is, therefore defined as follows:

$$h_q : \mathbb{S}^{\mathcal{X}} \longrightarrow [0, 1]$$

where $\mathbb{S}^{\mathcal{X}}$ denotes the universe of possible samples from $\mathcal{X}$.

Although quantification and classification tasks have a great similarity, researchers have devoted much more effort to develop the latter. However, in the last decade, quantification has been rediscovered by an active part of the machine learning community. Efforts have been made to formalize and to develop this task. [González et al. \(2017a\)](#) organized a comprehensive survey about quantification, providing, among other things, a taxonomy of quantification methods described by three different groups:

- I. *Classify, Count, and Correct*: methods that classify each instance firstly and then count how many belong to each class. Methods that apply any correction to their predictions are included in this group as well;
- II. *Traditional classification algorithms adaptations*: algorithms that modify the mechanics of traditional classification learning methods so that they become quantifiers;
- III. *Distribution matching*: algorithms that parametrically model the distribution of the training set and later search the parameters that produce the best match against the test data.

In our experiments, we compared quantifiers from first and third taxonomic groups to provide a significant degree of diversity to our analysis. Quantifiers from the second group do not use a traditional classifier in previous steps; they modify the whole classification learning algorithm otherwise. We next describe the principal quantification methods in the literature.

6.3 Brief survey of quantification methods

In this section, we provide a brief survey of some of the most prominent quantification methods available in the literature.

Classify and Count (CC) is arguably the simplest quantification method that derives from classification ([FORMAN, 2005](#)). After inducing a classifier, h from a training set T_r , each instance in the test set T_e is classified, and the proportion of the positive class is estimated by counting the positive classifications. Formally, CC is defined as:

$$CC_{\oplus}(T_e) = \frac{|\{\vec{x}_j \in T_e | h(\vec{x}_j) = \oplus\}|}{|T_e|}$$

Adjust Classify Count (ACC) acknowledges possible misclassifications and applies a correction to the count produced by CC. Given estimates for true positive (tpr) and negative (fpr) rates, usually estimated via cross-validation with the training set, ACC is defined as follows (FORMAN, 2005):

$$ACC_{\oplus}(T_e) = \frac{CC_{\oplus}(T_e) - fpr_{T_r}}{tpr_{T_r} - fpr_{T_r}} \quad (6.1)$$

Probabilistic Classify and Count (PCC) assumes that calibrated scores may have richer information than discrete decisions. The positive ratio is the average of such scores, as formalized by Gao and Sebastiani (2016):

$$PCC_{\oplus}(T_e) = \frac{1}{|T_e|} \sum_{\vec{x}_j \in T_e} \hat{\Pr}(\oplus|\vec{x}_j)$$

Similarly to CC, PCC underestimates or overestimate the actual prevalence when the distribution of the classes changes (TASCHE, 2014).

Probabilistic Adjust Classify and Count (PACC) consists of applying a correction rate to the PCC estimation. This adjustment is the same made by ACC to CC. Proposed by Bella *et al.* (2010) and named Scaled Probability Average (SPA). It can be expressed as:

$$PACC_{\oplus}(T_e) = \frac{PCC_{\oplus}(T_e) - fpr_{T_r}}{tpr_{T_r} - fpr_{T_r}}$$

Threshold selection methods similarly to classification, some quantification algorithms are affected when the class distribution in the training set is imbalanced. In the case of ACC, class imbalance leads to poor estimation of tpr and fpr that incur inordinate quantification corrections. To mitigate this situation, Forman (2006) introduced policies to guide threshold selection:

X selects the threshold where for which $fpr = 1 - tpr$;

Max selects the threshold that maximizes $tpr - fpr$;

T50 selects the threshold for which $tpr = 50\%$;

Median Sweep (MS) estimates true positive rates tpr and false positive rates fpr for a range of thresholds. The positive prevalence is estimated as the median of the predictions issued by ACC for each threshold.

HDy is a method based on distribution matching proposed by González-Castro, Alaiz-Rodríguez and Alegre (2013). The idea is to modify the distribution of training scores to fit the distribution of test scores. HDy-LP requires two normalized histograms of scores, P^+ and P^- , generated by a scorer s with the training set. P^+ depicts the distribution of scores

of positive examples, and P^- of negative ones. Given a test sample, a histogram $Q(T_e)$ estimates the distribution of scores obtained by s with T_e . The positive class ratio of the test set is estimated as:

$$\text{HDy}_{\oplus}(T_e) = \arg \min_{0 \leq \alpha \leq 1} \{ \text{HD}(\alpha P^+ + (1 - \alpha) P^-, Q(T_e)) \}$$

where HD is the Hellinger distance. The arg min translates to a linear search through several possible values of α . In (GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013), the authors estimate the positive ratio for varying numbers of bins: from 10 to 100 with increments of 10. The final positive class ratio reported is the median of the estimates.

DyS framework Maletzke *et al.* (2019) argue that HDy is an instance of a more general framework, where the positive class ratio is estimated as follows:

$$\text{DyS}_{\oplus}(T_e) = \arg \min_{0 \leq \alpha \leq 1} \{ \text{DS}(\alpha H(S^+) + (1 - \alpha) H(S^-), H(S(T_e))) \}$$

where DS is a dissimilarity function, S^+ , S^- , $S(T_e)$ are the sample of scores for the positive training data, negative training data, and test sample, respectively, and H is a function that converts a sample of scores into a representation that is compatible with DS and that supports mixing two distributions according to a factor α . Additionally, differently from HDy, in DyS, the arg min translates to a ternary search to speed up the algorithm.

The methods shown in this section present vastly different approaches, all of them rely on the assumption that the only change from training to test data is in $\text{Pr}(\oplus)$, that is, the prior probability of the classes.

For many applications, while it is reasonable not to expect substantial changes to the posterior probabilities, such an assumption implicates in a severe reliance on the training data being more representative than it might be. For instance, data collection bias may emphasize certain subgroups of the classes, in data training data, that may appear less frequently in practice. Additionally, subclasses that were not differentiated may be observed at varying rates at a later time, contributing to unexpected changes to the posterior probabilities. Such changes are prone to occur in virtually any application. This raises the question of how well the quantification algorithms can perform when facing such scenarios.

6.4 Proposed methods

We open this section by introducing a simplified parametric model used to simulate different score distributions. Such a model is useful to measure how quantifiers are impacted by changes in posterior probabilities, such as class separation and overlap, from training to test.

We close the section with the proposal of a new quantification algorithm that is more tolerant posterior probability changes than traditional quantification methods.

6.4.1 Model for Score Simulation

Scorers produce numerical values that correlate to $\Pr(\oplus|\vec{x})$. As such, changes in the posterior probabilities are directly reflected by changes in the distribution of scores. In a simplified view, when a scorer models the data accurately and the classes are fairly separated, positive scores should amass close to one, and negative scores close to zero. As the separation of the classes decreases or the scorer stops modelling the distributions adequately, the overlap between scores increases. In general, we can analyze the behaviour of methods that rely on scores by directly changing scores instead of manipulating the underlying data distribution.

We propose the **Model for Score Simulation (MoSS)** to produce synthetic scores simulating distinct overlap scenarios of score distributions. Our model is inspired by models used in (TASCHE, 2019) and (KAWAKUBO; PLESSIS; SUGIYAMA, 2016) to simulate the variability in feature distributions. However, we are interested in evaluating only the score complexity in binary quantification problems. MoSS takes one parameter, the *merging* factor m , which varies from zero to one and condensates the difficulty of the underling classification task. When $m = 0$, all positive observations have score one, while all negative observations have score zero, turning classification perfect and, as a consequence, quantification a trivial task. On the other hand, when $m = 1$, all scores for both classes are uniformly distributed within the interval $[0, 1]$, turning it impossible to distinguish between classes. In the equation below, we define the creation of a random sample containing n observations such that $0 \leq \alpha \leq 1$ is the proportion of the positive class.

$$\text{MoSS}(n, \alpha, m) = \text{syn}(\oplus, \lfloor \alpha n \rfloor, m) \cup \text{syn}(\ominus, \lfloor (1 - \alpha)n \rfloor, m)$$

where

$$\begin{aligned} \text{syn}(\oplus, n, m) &= \bigcup_{i=1}^n \{X_i^m\} \quad , \quad X_i \sim U(0, 1) \\ \text{syn}(\ominus, n, m) &= \bigcup_{i=1}^n \{1 - X_i^m\} \quad , \quad X_i \sim U(0, 1) \end{aligned} \tag{6.2}$$

A synthetic score in MoSS is a non-linear map of a uniformly distributed random variable in the range $[0, 1]$ to a non-uniform random variable also in the range $[0, 1]$. This implies that to calculate the density of the mapped random variable, that is, the proportion of the scores produced by MoSS in the range $[u, v]$, $0 \leq u < v \leq 1$, we can simply measure the length between

the inverse-map of u and v , as formalized below:

$$\begin{aligned}
 \text{syn}_{\text{density}}(\oplus, u, v, m) &= v^{\frac{1}{m}} - u^{\frac{1}{m}} \\
 \text{syn}_{\text{density}}(\ominus, u, v, m) &= (1 - u)^{\frac{1}{m}} - (1 - v)^{\frac{1}{m}} \\
 \text{MoSS}_{\text{density}}(\alpha, u, v, m) &= \alpha \text{syn}_{\text{density}}(\oplus, u, v, m) \\
 &\quad + (1 - \alpha) \text{syn}_{\text{density}}(\ominus, u, v, m)
 \end{aligned} \tag{6.3}$$

Equation 6.3 is not defined for $m = 0$, which represents a trivial case since all positive observations are one and all negative observations are zero. The well-defined density function implies that histograms for MoSS scores can be estimated in $O(b)$, where b is the number of histogram bins, without even synthesizing actual scores. Indeed, the density of any sample produced by MoSS with a significant enough number of observations converges to the one defined in Equation 6.3.

We can apply the same consideration regarding the base uniform distribution to synthesize scores without sampling random uniform values. Arguably, by removing randomness, we can generate small samples that are more representative of the distribution than otherwise. In this case, we redefine syn , used in MoSS, as follows:

$$\begin{aligned}
 \text{syn}'(\oplus, n, m) &= \bigcup_{i=1}^n \left\{ \left(\frac{i}{n+1} \right)^m \right\} \\
 \text{syn}'(\ominus, n, m) &= \bigcup_{i=1}^n \left\{ 1 - \left(\frac{i}{n+1} \right)^m \right\}
 \end{aligned}$$

6.4.2 Distribution y-Similarity using Synthetic Scores

Our second proposal is a quantification algorithm named Distribution y-Similarity using Synthetic Scores (DySyn), designed to be more tolerant against changes in posterior probabilities than traditional methods. DySyn is a combination between our synthetic scorer model and DyS framework proposed by [Maletzke et al. \(2019\)](#).

DyS searches for the best mixture between positive and negative score distributions estimated from the training set, aiming to find a value of α that minimizes the similarity between the mixed distribution and the test set distribution.

In DySyn, we replace the training scores with scores generated by MoSS. Although MoSS scores are generic and do not approximate any particular score distribution, we argue that the shapes they provide are suitable enough in the context of the search mechanism of DyS.

Additionally, in DySyn we not only search across multiple values of α via ternary search but also across different overlaps between positive and negative scores, by linearly varying the parameter m of MoSS. We argue that, for some value of m , the score distribution will be close

enough to real data that we can accurately measure the proportion of the positive class. DySyn is illustrated in Figure 41.

Figure 41 – The objective of DySyn is to search the synthetic distribution that is the most similar to the distribution of test scores. The corresponding α is reported. The blue shade corresponds to the contribution of positive scores, and red shade to the contribution of negative scores.

Source: Elaborated by Denis dos Reis.

As DySyn do not rely on a fixed distribution derived from training scores, it is unaffected by changes in the distribution, contrary to DyS. DySyn is formally defined as follows:

$$\text{DySyn}_n(Z) = \underset{\substack{0 \leq \alpha \leq 1 \\ \text{s.t. } 0 \leq m \leq 1}}{\arg \min} \{ \text{DS}(H(\text{MoSS}(n, \alpha, m)), H(Z)) \} \quad (6.4)$$

where Z is the scores of the test sample, m is the mixing factor which defines the overlapping degree between scores of positive and negative examples in the training data, and n is the number of synthetic scores. To get positive and negative training scores, we use the MoSS model with $\alpha = 1$ and $\alpha = 0$, respectively. We emphasize that for distance functions that adopt histograms, we need not generate synthetic scores and can derive histograms from Equation 6.3.

6.5 Experimental setup

In this chapter, we evaluate how several quantification methods perform under changes in the posterior probabilities of test data. We do so by proceeding with two experimental setups:

the first one adopts synthetic scores generated by MoSS and the second employs benchmark datasets.

The benefit of adopting synthetic scores produced by MoSS is that we can disregard the source of drift in the data, from training to test, and instead focus on the practical change to the scores they would cause. Notably, we analyze how quantifiers behave when the underlying classification task becomes easier or more difficult, in the test data, than it was in the training data.

In our first experimental setup, we adopted 0.5 as a classification threshold for methods that rely on predicted labels considering that, as m is the same for both classes, scores produced by MoSS are symmetric around this value for $\alpha = 0.5$. DySyn was excluded from this first analysis since it also utilizes MoSS and it would be at an unfair advantage against other methods. We varied each experimental parameter independently, as follows:

- α : the positive class proportion from 0% to 100% with increments of 1%;
- m_{Tr} the mixing factors for training scores, from 0.05 to 0.95 with increments of 0.05;
- m_{Te} : the mixing factors for test scores, from 0.05 to 0.95 with increments of 0.05
- n_{Tr} and n_{Te} : the size of training and test samples are fixed at 2,000 (1,000 for each class) and 100, respectively;
- the number of repetitions for each combination of the previous parameters is 10.

In our second experimental setup, we adopted benchmark datasets to achieve a more realistic scenario. However, in this setup, we only evaluated posterior probability changes caused by alterations in the proportion of subclasses (which are unknown to the quantification system). For the binary quantification purpose, we employed binarized multi-class datasets obtained from UCI (DHEERU; TANISKIDOU, 2017) and OpenML (VANSCHOREN *et al.*, 2013) repositories. Table 12 briefly describe the datasets. Specific citations are requested for Avila (STEFANO *et al.*, 2018) and Walking (CASALE; PUJOL; RADEVA, 2012). Additionally, we note that Jock A. Blackard and Colorado State University preserve copyright over Coverttype.

In each binarized dataset, the negative class comprises two subclasses. One has a more apparent separation from the positive class, while the other is more overlapping. In Table 12, the last two columns represent the overlap between the scores of each negative subclass and the positive class.

We uniformly split each dataset into two parts: training and test. We use a stratified sampling, including the negative subclasses. With the training part, we performed 10-fold cross-validation to obtain the classification scores used by the mixture models that follow the DyS framework, as well as to calculate the *tpr* and *fpr* performance statistics used in the variations of

Table 12 – Datasets description.

ID	Dataset	Size	Classes	Overlap (%)	
				Subclass 1	Subclass 2
A	Mosquitoes	24,000	3	2.84	26.25
B	Avila	20,867	12	0.29	5.97
C	Walking	149,332	22	1.32	17.28
D	Satimage	6,430	6	0.22	20.49
E	Coverttype	110,393	7	4.75	20.52
F	MFeat	2,000	10	2.66	20.18

Source: Research data.

ACC. The full training half was also used to induce a single scorer that was applied on the test half to obtain the test set scores. We produced all scores using Random Forests with 200 trees. For PCC and PACC methods we calibrate random forest scores, using an isotonic regression calibration based on pair-adjacent violators (NAEINI; COOPER, 2016).

We sampled random test sets composed of 100 instances from the test part. We varied the positive class proportion from 0% to 100% with increments of 5%. For each positive ratio, we varied the proportion of the first subclass within the negative portion from 0% to 100% with increments of 25%. We repeated each setting 10 times and averaged the results.

We measured the performance of quantifiers in our experiments in terms of Mean Absolute Error (MAE) (SEBASTIANI, 2018). MAE is the average of absolute differences between actual ($\vec{q}$) and predicted ($\hat{\vec{q}}$) quantification for a set of classes $\mathcal{C}$. We compared the methods according to the Friedman test with 95% confidence level and Nemenyi post hoc test.

Both DyS and DySyn have two relevant parameters: the dissimilarity function and the number of bins used to represent a sample of scores. Studies performed by Maletzke *et al.* (2019) provide a ranking of several dissimilarity functions for DyS algorithm as well as a recommended number of bins. In our experiments, we selected the top five similarity functions: ORD, Sample ORD (SORD), Probabilistic Symmetric (PS), Jensen Difference (JD), and Topsøe (TS). The positive class ratio reported by DyS and DySyn is the median for a varying number of bins, from 2 to 20 with increments of 2. We also include a new variation of HDy with Laplace smoothing (HDy-LP), making this more reliable on extreme scenarios.

Specifically for DySyn, we varied m from 0.1 to 0.9 with increments of 0.2. When using distance functions that require histograms, we applied Equation 6.3 to produce a histogram without generating scores. In the case of SORD, which is a sample distance function, we generated 1,000 scores for each class according to Equation 6.2.

6.6 Experimental evaluation

We open this section with our first experimental setup, which uses synthetic scores generated by MoSS. The intent is to analyze the behaviour of several quantifiers when the problem difficulty changes from training to test. Although we varied both m_{Tr} and m_{Te} in a wide range independently, due to space constraints we display plots of MAE against m_{Te} for only four fixed values of m_{Tr} . Additional results, as well as code, are available in our supplemental material website¹.

Figures 42 (a), (b), (c), and (d) show the mean absolute error when m_{Tr} is 0.05, 0.25, 0.55, and 0.75, as we vary m_{Te} . MAX presented a relatively poor performance for every setting. In all cases, CC and PCC displayed a steady increase of MAE as m_{Te} raised. The lack of adjustment can explain this behaviour: they depend solely on how easy it is to classify the instances. On the other hand, for all other algorithms, the smallest quantification error generally occurred when class separation in training and test was the same or very close, that is, when $m_{Tr} \approx m_{Te}$. These results are sound since such algorithms rely on the training and test sets having the same probability distribution.

DyS and HDy-LP explicitly model the distribution of scores, while ACC, PACC, X, T50, and MS estimate error rates from the training set, which are later used to adjust quantification. Estimates from a different distribution are likely to result in either too aggressive or too moderate adjustments. Notably, these algorithms displayed worse performance even when the separation of classes increases and the problem becomes easier ($m_{Te} < m_{Tr}$), restricting the practical use of the algorithms.

In the second part of our experimental evaluation, we compared our proposal against traditional and well-known quantifiers, in benchmark datasets. Table 13 shows the quantification performance of all evaluated quantifiers, averaging all variations of class and subclass proportions. Standard deviations are shown in Table 14. The best quantification result for each dataset is shown in bold. Underlined values represent a statistical difference against other methods or our proposal. For all datasets except Coverttype, our proposal outperformed other quantifiers.

Figure 43 shows the ranking for all variations of m_{Tr} and m_{Te} , simulating scenarios where the test set becomes harder and easier than the training set. We can see that the second quartile of DySyn-TS is below than the first quartile the second best-ranked algorithm. Indeed, we confirm the superiority of our proposal when the proportion of negative subclasses change from training to test.

Regarding the behaviour of algorithms, our results with benchmark datasets agreed with our first experimental setup, which adopted synthetic scores: methods that strictly rely on the distribution of training data performed worse when test data mismatched training data. Our experiments exposed how sensitive the existing quantification algorithms are to changes in the

¹ <https://quantification.shinyapps.io/dysyn/>

Figure 42 – Mean absolute error when m_{Tr} is 0.05, 0.25, 0.55, and 0.75, as we vary m_{Te} . Vertical lines indicate $m_{Tr} = m_{Te}$.

(a) Fixed $m_{Tr} = 0.05$.

(b) Fixed $m_{Tr} = 0.25$.

(c) Fixed $m_{Tr} = 0.55$.

(d) Fixed $m_{Tr} = 0.75$.

Source: Elaborated by the author.

Table 13 – Mean absolute error for each dataset and quantifier averaged over all negative subclasses distributions.

Quantifier	A	B	C	D	E	F
ACC	0.062	0.034	0.061	0.065	0.051	0.061
CC	0.055	0.035	0.069	0.039	0.088	0.044
DyS-TS	0.046	0.025	0.050	0.035	0.064	0.045
DySyn-TS	0.034	0.022	0.041	0.022	0.078	0.030
DySyn-SORD	0.042	0.028	0.048	0.038	0.071	0.033
HDy-LP	0.063	0.047	0.064	0.043	0.082	0.059
MAX	0.060	0.035	0.067	0.059	0.076	0.060
MS	0.089	0.054	0.078	0.086	0.061	0.076
PACC	0.045	0.026	0.060	0.053	0.081	0.039
PCC	0.094	0.049	0.098	0.069	0.136	0.106
SORD	0.060	0.037	0.061	0.058	0.058	0.055
T50	0.062	0.033	0.061	0.065	0.050	0.061
X	0.055	0.033	0.055	0.049	0.057	0.061

Source: Elaborated by the author.

Table 14 – Standard deviation of values presented in Table 13.

Quantifier	A	B	C	D	E	F
ACC	0.006	0.001	0.014	0.009	0.012	0.003
CC	0.002	0.002	0.004	0.005	0.007	0.004
DyS-TS	0.019	0.007	0.034	0.019	0.038	0.010
DySyn-TS	0.005	0.003	0.005	0.003	0.009	0.005
DySyn-SORD	0.010	0.007	0.013	0.003	0.013	0.007
HDy-LP	0.014	0.004	0.030	0.012	0.041	0.022
MAX	0.019	0.009	0.026	0.027	0.032	0.015
MS	0.020	0.016	0.025	0.015	0.022	0.009
PACC	0.020	0.004	0.037	0.023	0.018	0.014
PCC	0.006	0.004	0.010	0.007	0.007	0.004
SORD	0.012	0.011	0.024	0.014	0.024	0.008
T50	0.006	0.001	0.014	0.009	0.011	0.003
X	0.008	0.007	0.017	0.005	0.019	0.013

Source: Elaborated by the author.

posterior probability, including changes in the proportion of subclasses. In Figure 44 we present MAE for Walking, one of the datasets in which the effect was more evident. Additional raw results and plots are available in our supplemental material website.

We emphasize that our proposal is still competitive even when there are no changes in the posterior probability from training to test, as shown in Figure 45. This fact indicates that indiscriminate use of DySyn when there is uncertainty regarding possible changes is still beneficial.

Figure 43 – Aggregation of several rank positions for all quantification algorithms in all tested settings.

Source: Elaborated by the author.

Figure 44 – Mean absolute error for different ratios of the first subclass. For each ratio of the first subclass, the positive class ratio was also varied from 0% to 100% with increments of 5%. The vertical line indicates where training and test have the same negative subclasses distribution.

Source: Elaborated by the author.

6.7 Conclusion

This article is the first to show how binary quantifiers are affected when score quality vary from training to test. We include in our experiments the most used and popular binary quantifiers that adopt a base scorer or classifier. We illustrate the shortcomings of quantifiers in three different situations: (i) when the distribution of the scores obtained from the test set is equal to the training set; (ii) when scores of positive and negative classes have higher overlap in

Figure 45 – Aggregation of several rank positions for all quantification algorithms. The unknown negative subclasses posterior distribution matches the training distribution.

Source: Elaborated by the author.

test data than training data, that is, the problem becomes harder after deployment; and (iii) when the score distribution has a lower overlap than the observed in training set, that is, the problem becomes easier. As expected, all quantifiers had their performance degraded when the problem becomes harder in test data. However, and more interestingly, all quantifiers excepting Classify and Count also were negatively impacted when the problem becomes easier.

To circumvent such changes, we proposed DySyn. It is a reliable quantifier that can function under variability of the score distribution. We compared 11 quantification methods representative of different counting paradigms with six benchmark datasets and confirmed the superiority of our method.

Future efforts include the development of new methods that can function when the posterior probability of data changes from training to test. Although this setting is very challenging, it is also more realistic, as two sources of such issue, named data collection bias and unknown subclasses, are common problems in machine learning applications.

OTHER CONTRIBUTIONS

The main topic of this thesis is binary quantification in more realistic scenarios. Although we have described our contributions to this topic, we also collaborated on other related subjects and applications. An example is a new proposal to deal with concept drift, describing the data stream as a set of contexts, building a specific classifier for each context. Therefore, the challenge is to identify and use the appropriate model rather than only one single model that was induced with all available training data (REIS; MALETZKE; BATISTA, 2018). As result, the proposal has obtained a performance close to that of a *topline* that always use the correct classification model and higher than the single model induced with all available training data.

Following that idea, in (REIS *et al.*, 2018), we also contributed with an upgrade of the proposal mentioned above for dealing with different contexts. In this new proposal, we use a mixture model quantification algorithm as background in to newly proposed methods (SMR-HDy and XO-HDy), aiming to identify contexts accurately even when the proportions of classes change. After that, we extended one of the methods presented in (REIS *et al.*, 2018) to develop our drift detector that is insensitive to class imbalance. The proposed drift detector also provides as a byproduct an estimate of the class distribution in the test set, which can be used to dynamically adjust the classifier's threshold.

Another contribution in the quantification area is the first formalization of the one-class quantification task on the open-set scenario. To the best of our knowledge, literature has mostly ignored the task of open-set quantification. In (REIS *et al.*, 2018), open-set quantification is defined and formalized. We also show that our methods can learn exclusively from positive class examples, and are more accurate when facing unseen classes than the state-of-the-art.

In the next section, we present our contribution to building an insect database involving several species and the first experiment with an intelligent trap that captures insects selectively.

7.1 Flying insect dataset

As explained in Chapter 1, during the development of this research we were interested in counting flying insects. Thus, in the last years, we have gathered data from different species in collaboration with other researchers and laboratories.

To build an insects dataset, we have collected data using the optical sensor briefly described in Chapter 1. The gathering process was done in São Carlos, SP, Brazil (University of São Paulo), Riverside, CA, United States of America (ISCA Technologies), and Raleigh, NC, United States of America (North Carolina State University). To control the species and sex, and therefore obtain labeled data, we used a small cylindrical chamber with the optical sensor embedded (Figure 46).

Figure 46 – Cylindrical chamber with the optical sensor embedded.

Source: Souza *et al.* (2019).

Moreover, some environmental conditions affect the behavior of insects even during the flight, such as humidity, altitude, wind speed, and temperature (VILLARREAL; WINOKUR; HARRINGTON, 2017; BELTON; COSTELLO, 1979). Air temperature has been the most relevant and well-known variable that impacts the insects' wing-beat frequency (WBF). Figure 47 shows the wing-beat frequency variability for different air temperatures.

Therefore, aiming to control the main influencing variable, we use chambers where the temperature and humidity are manually adjusted using a custom circuitry, as illustrated in Figure 48. Thus, to collect data from each species, we varied the temperature from 22°C to 34°C (increments of 2°C) and the humidity from 20% to 90%, considering different combinations of both variables. As mentioned in Chapter 1, when a flying insect inside each cylindrical chamber crosses the optical sensor's line-of-sight, variations are observed and converted in an audio signal.

Figure 47 – Influence of temperature on the wing-beat frequency for *Aedes aegypti* (female) mosquitoes.

Source: Souza *et al.* (2019).

At the same instant, temperature, humidity, luminosity, altitude, and air pressure are recorded as well. An example of audio captured using a cylindrical chamber is showed in Figure 49.

We collected more than one million instances for 20 different insect species including mosquitoes, houseflies, bees, and wasps. For 10 out of 20 insect species, it was possible to collect the data for each individual sex, totaling 30 class labels. Figure 50 shows the distributions of each species collected.

Our main focus was gathering data from different disease vectors, but we also included agricultural pests and pollinators. A brief description of insect that is a disease vector is presented below:

***Aedes aegypti*:** in Brazil, it is well-known as the dengue fever mosquito, but it also transmits diseases such as yellow fever, chikungunya, Zika fever, Mayaro, among other. This mosquito originated in Africa and was first identified in 1900 as a yellow fever vector. This mosquito is a global concern and it is present in tropical, subtropical and temperate regions around the world (SOUZA-NETO; POWELL; BONIZZONI, 2018);

***Aedes albopictus*:** this mosquito is an efficient disease vector that occurs in the tropical and subtropical regions of Southeast Asia, the Pacific, and most Indian Ocean islands (SCHOLTE *et al.*, 2008). Also known as Asian tiger mosquito or forest mosquito, it is a transmitter of 22 arboviruses, including West Nile virus, dengue virus, Japanese encephalitis virus, Chikungunya virus, Rift Valley fever virus, and Ross River virus;

***Aedes japonicus*:** originated in Japan and Korea, this mosquito has been reported in parts of North America and Central Europe. According to Schaffner *et al.* (2009), it is a competent

Figure 48 – Chamber used to control temperature and humidity.

(a) Chamber used at University of São Paulo to control temperature and humidity.

Source: [Souza et al. \(2019\)](#).

(b) Chamber used North Carolina State University to control temperature and humidity.

Source: Elaborated by the author.

Figure 49 – A signal gathered when an *Aedes aegypti* (female) crosses the line-of-sight. Source: Souza *et al.* (2019).

Figure 50 – Distribution of each species.

Source: Elaborated by the author.

arboviruses vector, including the West Nile virus;

***Anopheles aquasalis*:** this mosquito is the primary vector of Malaria disease, infecting around 400,000 persons each year. *Anopheles aquasalis* populations are usually sited in tropical and subtropical regions, requiring elevated temperature and humidity (CARVALHO *et al.*, 2002);

***Anopheles quadrimaculatus*:** it is a common malaria vector that is presently found in eastern

North America. Malaria was a threatening disease in the United States until its eradication in the 1950s, in spite of the continued presence of its transmitter;

Culex quinquefasciatus: this mosquito is the primary urban and semi-urban disease vector in Asia, responsible for the transmission of Lymphatic filariasis, and West Nile virus in North America. Recently, transmission of Zika virus has also been associated with this species in Brazil and China (GUEDES *et al.*, 2017);

Culex tarsalis: although *Culex quinquefasciatus* is one of the most relevant arboviruses vectors, *Culex tarsalis* is known as the Western Encephalitis mosquito in North America. This mosquito can be found in almost every region within the U.S, spreading mainly by human activity;

Phlebotomus papatasi: Leishmaniasis is a severe disease causing the death of more than 50,000 people annually. The *Phlebotomus papatasi* is the transmitter of *Leishmania* protozoan parasites and Brazil is between the most affected countries.

Figure 51 shows the distribution of all data gathered over both temperature and humidity variables. The gray bar represents the number of audio data recorded given a pair of humidity and temperature values. Our data collection has contemplated a wide range of combinations, with more instances when humidity is around 80%.

Figure 51 – Distribution of our data collection over both temperature and humidity.

Source: Elaborated by the author.

Sounds produced by insects, especially mosquitoes, have been studied over the years with results reported since the mid-20th century (CHADWICK; WILLIAMS, 1949; VILLARREAL; WINOKUR; HARRINGTON, 2017; SALIM *et al.*, 2017). However, only in the last decade mechanisms have been designed to identify different species of mosquitoes using the sound (GENOUD *et al.*, 2018; BATISTA *et al.*, 2011). In this sense, characteristics of mosquito sounds, primarily the wing-beat frequencies (WBF), are extensively studied in literature with a straightforward application in species identification (ARTHUR *et al.*, 2014).

There are more than 3,000 mosquito species in the world and many more flying insect species. The fundamental frequency (or wing-beat frequency) is one of the most relevant characteristics from the sound to identify some species. Figure 52 shows distributions of WBF for 20 species, including those where the sex was collected separately, thus totaling 30 labels. Although most of the mosquito species have a well defined WBF distribution, it is evident that using only this attribute alone is not a definite solution for species identification. In this figure, we can note a significant overlap between the species, depicting the complexity of the application. As mentioned, the data gathering process produces signals that are very similar to an audio record. Therefore, signal processing techniques can provide many other features aiming to construct a clearer separation between the species.

Figure 52 – Density functions of different insect species.

Source: Elaborated by the author.

Moreover, we are aware of data collection bias. For instance, the number of instances gathered in each temperature varied independently for each species and it also happened with other conditions. Thus, gathering data for this application is a challenging and an endless process.

Our data-gathering efforts have contributed directly and indirectly with other researches and also receive contributions from other researchers. We list below publications that summarize the relevance of our data gathering process:

- SOUZA, V.; REIS, D.; MALETZKE, A.; BATISTA, G. Challenges in benchmarking

stream learn-ing algorithms with real-world data. Submitted to *Data Mining and Knowledge Discovery*, p. 1–59, 2019.

- NADAI, B.L.; MALETZKE, A.G.; CORBI, J.J.; BATISTA, G.E.A.P.A.; REISKIND, M. "Wingbeat frequency variability captured by optical sensors from the mosquito vector *Aedes aegypti*". Submitted to *PLoS neglected tropical diseases*, 2019.
- NADAI, B.; MOURA, L.; MALETZKE, A.; BATISTA, G.; CORBI, J. "Variability of optical sensors used in the study and monitoring of *Aedes aegypti* mosquito". In: Proceedings of the Congresso Brasileiro de Medicina Tropical, 2019.
- SILVA, L.T.; SOUZA, V. M. A.; BATISTA, G. E. A. P. "EmbML Tool: supporting the use of supervised learning algorithms in low-cost embedded systems". In: Proceeding of the International Conference on Tools with Artificial Intelligence, 2019.
- SILVA, L. T.; SOUZA, V. M. A.; BATISTA, G. E. A. P. A. "Uma Ferramenta de Suporte ao Uso de Classificadores em Sistemas Embarcados". In: Proceeding of the Simpósio Brasileiro de Automação Inteligente, 2019.
- MALETZKE, A.G.; MILARE, C; NADAI, B.L.; SAROLI, J.; SINGH, S.; CORBI, J.J.; MAFRA-NETO, A.; KEOGH, E.; BATISTA, G.E.A.P.A. "Automatic insect recognition with optical sensors with variability of temperature and humidity". In: Proceedings of the AMCA 84th Annual Meeting, 2018.
- COELHO, B.; MALETZKE, A.; BATISTA, G.; "Optimizing classifier parameters for insect datasets". In: Proceedings of the XXVI USP International Symposium of Undergraduate Research, 2018.
- JETSMEN, Z.; MALETZKE, A.; BATISTA, G. "Evaluation of pitch detection methods for insect classification". In: Proceedings of the XXVI USP International Symposium of Undergraduate Research, 2018.
- NADAI, B.; BATISTA, G.; CORBI, J. "Estudo da frequência do batimento das asas do mosquito *Aedes aegypti* utilizando sensores inteligentes: simulação de diferentes temperaturas em laboratório". In: Proceedings of the XXXII Congresso Brasileiro de Zoologia, 2018.
- PARMEZAN, A. R. S.; SOUZA, V. M. A.; BATISTA, Gustavo E. A. P. A. "Towards Hierarchical Classification of Data Streams". In: Proceedings of the Iberoamerican Congress on Pattern Recognition, 2018.
- SOUZA, V. M. A.; ROSSI, R. G.; BATISTA, G.E. A. P. A.; REZENDE, S. O. "Unsupervised active learning techniques for labeling training sets: an experimental evaluation on sequential data". In *Intelligent Data Analysis*, v. 21, n. 5, p. 1061–1095, 2017,

7.2 Smart trap

Mosquito control is a global concern and an important strategy of interrupting transmission of several threatening diseases. Control efforts require an efficient surveillance system to assess control outcomes as well as identify areas with the highest risk of outbreaks. Surveillance methods are mostly based on trapping schemes, using traps composed of a specific lure to attract and capture mosquitoes that are seeking hosts or laying eggs. Different traps with distinct designs, lures, and costs were developed over the years (PEZZIN *et al.*, 2016; SANTOS *et al.*, 2019). Most of them are used to monitor and control adult mosquitoes and require a specialist to identify and count the mosquitoes trapped, which is an expensive and time-consuming task.

Usually, insect traps are divided into two groups: active and passive. The former uses movement to capture the insects, applying a fan induced airflow. Thus, it requires a power supply such as batteries, which must be charged and replaced. The sticky traps are a passive and inexpensive capture method. However, the surveillance remains expensive since it involves manual counting and classification. Ovitrap are another common passive trap that uses water to attract ovipositioning female mosquitoes to a chamber. Once in the chamber, the way out is more complicated than the process to get inside, keeping the mosquitoes trapped. Figure 53 shows examples of both active and passive traps.

In both active and passive traps, the species identification and counting are manual tasks, requiring a human expert. Manual identification, in its turn, produces estimations of mosquito populations with a large time delay. Therefore, a more time-responsive trap and less dependent on human intervention is desirable.

In this sense, in (SOUZA *et al.*, 2019) we propose a new trap design that selectively captures insects based on their species. Figure 54 shows a prototype of the Smart trap. This trap has the optical sensor embedded. Hence, it can sense the flying insect as it passes and only then turns a fan on to capture it. An interesting remark is that the fan is only activated if a target insect is detected. Otherwise, it reverts the fan direction, expelling the insect. This new design, named semi-active, has the capability of counting and classifying flying insects in real-time without the analysis of experts.

From the beginning of this research, we pursue to reach some contribution to the mosquito control challenge. Therefore, we spent our efforts on collecting data, proposing new binary quantification algorithms and performing several experiments on different sites. Next, we present one of this experiments using the Smart trap.

7.2.1 Experiments

We evaluate the Smart trap to answer two questions: (i) *how accurate is the trap to capture a target species and expelling another?* and (ii) *how accurate a quantifier is when both species are caught?* To answer both questions, we initially should learn a classifier to

Figure 53 – Examples of insect traps. (a) The ovitrap is a passive trap and (b) the BG-Sentinel2 an active trap which use light and carbon dioxide as lure.

(a) Ovitrap: passive trap.

Source: Souza *et al.* (2019).

(b) BG-Sentinel2: active trap.

Source: Souza *et al.* (2019).

Figure 54 – Smart trap with the optical sensor embedded.

Source: Souza *et al.* (2019).

embedded into the trap. For this reason, we use a subset of instances sampled from the dataset described in Section 7.1. This subset contains instances of *Aedes aegypti* and *Anopheles quadrimaculatus*, since they are the species involved in our trials¹. As discussed before, the temperature has an important role in the wing-beat frequency. Thus, we sample 1000 instances from each temperature range shown in Figure 47. We use a Support Vector Classifier with a Gaussian kernel (RBF) to induce the classifier that is embedded.

Initially, we performed a toy experiment in which we embedded a classifier into the cylindrical chambers. We use three chambers: in the first, we insert only female *Aedes aegypti*; the second one contains only female *Anopheles quadrimaculatus*; and the last one contains females of both species. We monitored this scenario for five hours in a controlled temperature of $28^{\circ}\text{C} \pm 3$ and humidity of $50\% \pm 10$. The results of this toy experiment allow us to evaluate if the classifier answers properly as well as provides an accuracy estimation.

From the results of this initial toy experiment, we performed a more complete test, simulating a semi-field conditions in the laboratory. To this end, we maintain a small population of mosquitoes composed of two different species in a controlled site (cage). Herein, we report the materials used:

Cage: for spreading mosquitoes in a controlled site, we built a small cage in a room with controlled temperature ($28^{\circ}\text{C} \pm 3$) and humidity ($50\% \pm 10$). The cage dimension consists

¹ The species used in our experiments were selected according to the availability

of $1.2\text{m} \times 1.2\text{m} \times 1.2\text{m}$ ($w \times l \times h$) made of a mesh screen and supported by a wood frame. Figure 55 shows the cage built. We performed this trial in the Grinnell's Laboratory at the North Carolina State University in Raleigh, North Carolina, United States of America, in February 2019;

Mosquitoes: in our trials, we only used gravid female mosquitoes from both *Aedes aegypti* and *Anopheles quadrimaculatus*. Mosquitoes of both species are from a colony established in the Biological Resources Facility (BRF) at North Carolina State University;

Lure: carbon dioxide (CO₂) is the most commonly used source of attractant. The release rate was 8 ml/min from a gas cylinder. The CO₂ is conducted through a pipe from the cylinder to the trap's chamber entrance.

Figure 55 – Cage used in the Smart trap trials.

Source: Elaborated by the author.

The first question is a classification problem where, for each mosquito that crosses the sensor, the classifier decides whether the capture mechanism should be activated or not. The efficiency of this experiment is related to the performance of the classifier and how responsive the trap is. Thus, we put the trap at the center of the cage and released 20 mosquitoes inside the cage. After 24 hours, we turned the trap off to verify how many mosquitoes of each species were

caught. We repeated this experiment varying the species distribution from 25% to 75%, with increments of 25%.

Although the second question could be evaluated using only the cylindrical chambers shown in Figure 46, we also investigate if the trapping mechanism affects the data recorded when each mosquito is caught. Thus, we removed the selective capture mechanism, *i.e.*, all mosquitoes that cross the sensor were captured and the features from the sound produced by each mosquito were extracted and recorded. At the end of the experiment, the batch of events was quantified to estimate the class distribution, *i.e.*, how many mosquitoes from each species were caught. Again, we put the trap at the center of the cage and then 20 mosquitoes were released inside the cage. We repeated this experiment 11 times, varying the species distribution from 0% to 100%, with increments of 10%. The duration of each experiment was 24 hours. After this time, we turned the trap off to verify how many mosquitoes of each species were caught. In this experiment, the quantifiers used adopted a Random Forest with 200 trees as a base classifier or scorer, maintaining consistency with setups used in previous chapters.

7.2.2 Results

We open this section showing the results of our preliminary experiment. Figure 56 shows the three cylindrical chambers used, containing female *Aedes aegypti* (left), female *Anopheles quadrimaculatus* (right), and females of both species (center).

Figure 56 – A simple experiment with a classifier embedded into the cylindrical chambers. The *left* chamber contains *Aedes aegypti* females, the *right* one *Anopheles quadrimaculatus* females, and the *center* chamber females from both species were put in.

Source: Elaborated by the author.

The digital displays in each chamber show accumulated classification result when a mosquito crosses the sensor. The classifier has reached accuracies of 100% for female *Aedes*

aegypti and 98% for female *Anopheles quadrimaculatus* at the end of the experiment. The chamber with mosquitoes from both species (*center*) was only included as an informative purpose, we are not able to verify the species of each mosquito that crosses in the light in this chamber.

Motivated by such initial results, we embedded the classifier into the trap to perform experiments in the cage. In the first part of our experimental evaluation, we assess the capturing mechanism along with with our previously tested classifier. Table 15 shows the results of this experiment.

Table 15 – Performance of the Smart trap to identify and capture a target species (*Aedes aegypti*).

Mosquitoes released		Captured		Accuracy
<i>Aedes aegypti</i>	<i>Anopheles quadrimaculatus</i>	<i>Aedes aegypti</i>	<i>Anopheles quadrimaculatus</i>	
5	15	5	0	100%
10	10	9	1	90%
15	5	12	2	86%

Source: Elaborated by the author.

Results in Table 15 only show part of the experiment. They report the results regarding the capture mechanism which works along with the classifier. Figure 57 shows the entire effort of the classifier to identify both target and non-target species. There was a high attempt rate from the female *Anopheles quadrimaculatus* to get into the trap after the CO₂, but the classifier presents a good response.

Figure 57 – Events recorded in the trap during each experiment, varying the species distribution.

Source: Elaborated by the author.

In our last experiment, without the selective trapping mechanism, we increase the number of repetitions, including trials with more variability of the species distribution. Our aim is to

evaluate the most popular counting algorithms, including our proposal DyS, from data recorded by the trap. From the data collected in each repetition, we apply the simple strategy Classify and Count (CC), Adjust Classify and Count (ACC), Median Sweep (MS), HDy, DyS using the distance function Topsøe. Table 16 shows the absolute error for each aforementioned quantifiers for all repetitions performed.

Table 16 – Absolute error of different quantifiers given the data recorded by the trap during the experiments in the cage.

Mosquitoes Captured		Target distribution	CC	ACC	MS	HDy	DyS+TS
<i>Aedes aegypti</i>	<i>Anopheles quadrimaculatus</i>						
0	9	1.00	0.00	0.00	0.00	0.00	0.00
1	15	0.94	0.06	0.06	0.00	0.06	0.06
3	15	0.83	0.06	0.06	0.06	0.06	0.06
5	8	0.62	0.15	0.15	0.08	0.15	0.15
7	7	0.50	0.00	0.00	0.00	0.07	0.07
10	8	0.44	0.06	0.06	0.06	0.11	0.06
12	6	0.33	0.06	0.06	0.06	0.11	0.11
14	5	0.26	0.05	0.05	0.05	0.11	0.05
14	1	0.07	0.00	0.00	0.00	0.07	0.00
18	2	0.10	0.00	0.00	0.00	0.05	0.00
20	0	0.00	0.00	0.00	0.00	0.00	0.00
Mean Absolute Error			0.04	0.04	0.03	0.07	0.05
Coefficient of Variation			1.19	1.19	1.17	0.66	0.98

Source: Elaborated by the author.

We cannot tell, only from this experiment, which quantifier is more relevant for mosquito population estimation and it also was not our purpose. We must keep in mind that the separation of the wing-beat frequency distribution between both species is well defined (Figure 52). Therefore, the simple classify and count method can perform sufficiently well for this problem against a more sophisticated method. However, both mixture models reported the lowest coefficients of variation, indicating more stability. In general, this behavior is expected, corroborating our results presented in Chapter 5, where we noted that MS and CC performed better than other approaches for small batches.

Summing up, this was the first experiment using a semi-active trap that captures insects based on decisions performed by a machine learning model. These first experiments help us to validate our initial ideas and provide new insights for future steps.

7.3 Collaborations in other areas

We also collaborated with other researchers in different areas. In the medicine field, we have developed tools to classify and measure pressure injuries. Pressure injuries are an injury to the skin and underlying tissue, causing considerable suffering and demanding an expensive treatment. Thus, our efforts consist in providing a computational system to assist professionals to care for patients by monitoring and classifying different levels of pressure injuries (TIBES *et al.*, 2017).

Another contribution to the medical field allied to time series mining is presented in (MALETZKE *et al.*, 2017). In this work, we present a method for building attributes in medical temporal databases, combining global and local features derived from different representations. Our proposal has enhanced the descriptive power for time series data and increased the performance of the models.

CONCLUSION

This work presents research efforts that led to improvements in quantification task. We initially envisioned quantification in a non-stationary environment, described as data streams. In that scenario, there are plenty of other constraints beyond those associated with our main task (quantification), such as changes in the feature space and label scarcity (SILVA *et al.*, 2018). Our first proposal, presented in Chapter 2, was an algorithm resilient against changes in the feature space, named SQSI (MALETZKE; REIS; BATISTA, 2017).

This initial result was important for showing that quantifying data streams by using a drift detection mechanism, allied with simple quantification techniques, produces better results than the simple classify and count strategy. However, our first proposal has a high dependency on labels when drift is detected, demanding class labels for all instances in the drifting batch. Our second contribution (SQSI-IS) (MALETZKE; REIS; BATISTA, 2018) reduces the dependency on actual labels when a drift is detected in comparison to former proposal. We have achieved that by applying active learning and semi-supervised approaches.

In both SQSI and SQSI-IS, we also embedded a simple mechanism aiming to be resilient under linear changes, *i.e.*, we try to correct each observed batch of data by applying a linear transformation. This mechanism is fairly simple but provides us even less dependency on class labels. SQSI-IS strategy allows us to reach the same or even better results while requiring less than 50% of labels required by SQSI when a drift is detected.

Our initial proposals have as main limitations the treatment of only linear distortions and the lack of memory mechanism to avoid requesting labels for concepts learned in the past. Reis *et al.* (2018) proposed an algorithm to tackle this issue with recurrent concepts. That proposal was evaluated on a classification scenario. SQSI-IS allied with that recurrent concepts mechanism seems to be a straightforward future step.

Although there is plenty of other quantifiers that can be adapted for a non-stationary scenarios, we realized in our experiments another more urgent demand. We noted a high rate of

false positives reported by drift detection mechanisms when batches have a high imbalance class distribution.

Therefore, we decide to assign some effort to mitigate the high false-positive rate. As a result, in Chapter 3 we propose an unsupervised drift detector which is insensible to class imbalance. This method is based on the HDy algorithm and a technique for learning with recurrent concepts as an unsupervised concept drift test (REIS *et al.*, 2018). Our results clearly show that our approach can detect feature changes while being insensitive to the distribution of the classes. Moreover, this proposal provides as a byproduct an estimate for the class distribution, which may be used to dynamically adjust the decision threshold of classifiers.

There are several drift detectors in literature. However, the majority of them were proposed for the classification task. Although those methods can be used for quantification with some adaptations, most of them require the labels to provide some decision. Our proposal is an unsupervised detector and we suppose it can be extended to work similarly to the well-known methods such as Drift Detection Method (DDM) and Early Drift Detection Method (EDDM) (BAENA-GARCIA *et al.*, 2006). This could be done without requiring labels and with more than one threshold embedded, defining warning and drift triggers.

Following this direction, we are working on an upgrade version of the method proposed by Reis *et al.* (2018). While the original method can identify concepts that are in a fixed set, it is unable to reckon that a test set has a yet unknown concept. In that sense, we intend on improving it by detecting when a new concept arrives. We are confident that this feat can be achieved by aligning CDT (MALETZKE *et al.*, 2018) and SMR-HDy (REIS *et al.*, 2018).

During our review of approaches proposed by González-Castro, Alaiz-Rodríguez and Alegre (2013) and Reis *et al.* (2018), which explore mixture models for quantification and context detection using HDy algorithm, we found out some unclear points, such as the dissimilarity function and the representation used to describe distributions. Although HDy is the state-of-the-art Mixture Model approach, we argued that it is not more than an instance of a more general and unexplored framework. In Chapter 4, we conducted an extensive experimental evaluation, including several dissimilarity functions and distribution representations. For the first time, an ordinal dissimilarity function was employed in a mixture model (ORD). ORD avoids being affected by such sparseness since the relation between neighbor dimensions of a histogram is considered, rather than each dimension contributing in isolation to the magnitude of the distance. Thus, we extend the state-of-the-art mixture model and proposed a new one, redefining the best dissimilarity function and distribution representation. Moreover, we speed up our proposal including a ternary search mechanism to find the best match between training and test distributions.

On the other hand, in our experiments, we noted a severe instability from all possible parameter settings for our proposed framework, including the well-known HDy algorithm. Thus, we decided to explore that instability in Chapter 5, including other quantifiers.

We pointed out that the literature has neglected a relevant aspect of real-world problems. Providing reliable quantification for small batches is an open question. In this sense, we gave our contribution to this question as well. We formalize a new experimental design to evaluate quantification algorithms comprehensively. We expect that future papers include in their evaluations the batch size variability, providing a more reliable ranking of quantifiers. However, we also offer an initial effort to deal with this question. We propose a metalearning scheme to select the best quantifier based on the test set size that can outperform the best single quantification method.

Chapter 6 is partially like a return to our initial question: *how to quantify under non-stationary conditions?* In this chapter, we tackle this question as a problem of facing changes in the scores' quality, which are the core of many quantifiers. We built various scenarios, indicating problems with different difficulty degrees. More importantly, we analyzed the behavior of state-of-the-art quantifiers trained with a given score distribution as such distribution changes. As expected, all quantifiers had their performance degraded when the problem becomes harder in test data. However, and more interestingly, all quantifiers excepting Classify and Count also were negatively impacted when the problem becomes easier. To circumvent such changes, we proposed DySyn. It is a reliable quantifier that can function under the variability of the score distribution.

At last, Chapter 7 describes other efforts in the quantification area, mosquito data gathering, mosquito trapping mechanisms, and collaborations in other areas. We performed the first experiment, in the laboratory, using an intelligent mechanism to capture mosquitoes selectively. This step put us in the cutting edge of the development of traps supported by machine learning.

BIBLIOGRAPHY

ARTHUR, B. J.; EMR, K. S.; WYTTEBACH, R. A.; HOY, R. R. Mosquito (aedes aegypti) flight tones: Frequency, harmonicity, spherical spreading, and phase relationships. **The Journal of the Acoustical Society of America**, ASA, v. 135, n. 2, p. 933–941, 2014. Citations on pages 27 and 141.

BAENA-GARCIA, M.; CAMPO-AVILA, J. del; FIDALGO, R.; BIFET, A.; GAVALDA, R.; MORALES-BUENO, R. Early Drift Detection Method. In: **Proceedings of the 4th ECML PKDD International Workshop on Knowledge Discovery from Data Streams**. [S.l.: s.n.], 2006. p. 77–86. ISBN 3-540-23237-0. ISSN 0302-9743. Citation on page 152.

BARRANQUERO, J.; GONZÁLEZ, P.; DÍEZ, J.; COZ, J. J. del. On the study of nearest neighbor algorithms for prevalence estimation in binary problems. **Pattern Recognition**, Elsevier Science Inc., New York, NY, USA, v. 46, n. 2, p. 472–482, Feb. 2013. ISSN 0031-3203. Available: <http://dx.doi.org/10.1016/j.patcog.2012.07.022>. Citations on pages 99 and 107.

BATISTA, G. E.; KEOGH, E. J.; MAFRA-NETO, A.; ROWTON, E. Sigkdd demo: sensors and software to allow computational entomology, an emerging application of data mining. In: **ACM. Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.], 2011. p. 761–764. Citations on pages 26 and 141.

BEIJBOM, O.; HOFFMAN, J.; YAO, E.; DARRELL, T.; RODRIGUEZ-RAMIREZ, A.; GONZALEZ-RIVERO, M.; GULDBERG, O. H. Quantification in-the-wild: data-sets and baselines. **arXiv preprint arXiv:1510.04811**, 2015. Citation on page 25.

BEKKER, J.; DAVIS, J. Estimating the class prior in positive and unlabeled data through decision tree induction. In: **Proceedings of the 32th AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2018. Citation on page 79.

BELLA, A.; FERRI, C.; HERNÁNDEZ-ORALLO, J.; RAMIREZ-QUINTANA, M. J. Quantification via probability estimators. In: **IEEE. 2010 IEEE International Conference on Data Mining**. [S.l.], 2010. p. 737–742. Citations on pages 37, 103, 107, and 123.

BELTON, P.; COSTELLO, R. A. Flight sounds of the females of some mosquitoes of western canada. **Entomologia experimentalis et applicata**, Wiley Online Library, v. 26, n. 1, p. 105–114, 1979. Citation on page 136.

BIFET, A.; GAVALDÀ, R. Adaptive learning from evolving data streams. In: **IDA**. [S.l.: s.n.], 2009. p. 249–260. Citation on page 64.

BRAZDIL, P.; GIRAUD-CARRIER, C.; SOARES, C.; VILALTA, R. Metalearning for algorithm recommendation: an introduction. **Metalearning: Applications to Data Mining**, Springer, p. 11–29, 2009. Citation on page 109.

CANDANEDO, L. M.; FELDHEIM, V. Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models. **Energy and Buildings**, Elsevier, v. 112, p. 28–39, 2016. Citations on pages 87 and 111.

CANDILLIER, L.; LEMAIRE, V. Design and analysis of the nomao challenge active learning in the real-world. In: **Proceedings of the ALRA: Active Learning in Real-world Applications, Workshop ECML-PKDD**. [S.l.: s.n.], 2012. Citations on pages 87 and 111.

CARLSON, C. J.; DOUGHERTY, E. R.; GETZ, W. An ecological assessment of the pandemic threat of zika virus. **PLoS neglected tropical diseases**, Public Library of Science, v. 10, n. 8, p. e0004968, 2016. Citation on page 26.

CARVALHO, S. C. G. d.; JUNIOR, A. d. J. M.; LIMA, J. B. P.; VALLE, D. Temperature influence on embryonic development of anopheles albitarsis and anopheles aquasalis. **Memórias do Instituto Oswaldo Cruz**, SciELO Brasil, v. 97, n. 8, p. 1117–1120, 2002. Citation on page 139.

CASALE, P.; PUJOL, O.; RADEVA, P. Personalization and user verification in wearable systems using biometric walking patterns. **Personal and Ubiquitous Computing**, Springer-Verlag, v. 16, n. 5, p. 563–580, 2012. Citation on page 128.

CHA, S.-H. Taxonomy of nominal type histogram distance measures. **City**, v. 1, n. 2, p. 1, 2008. Citation on page 82.

CHA, S.-H.; SRIHARI, S. N. On measuring the distance between histograms. **Pattern Recognition**, Elsevier, v. 35, n. 6, p. 1355–1370, 2002. Citation on page 83.

CHADWICK, L. E.; WILLIAMS, C. M. The effects of atmospheric pressure and composition on the flight of drosophila. **The Biological Bulletin**, v. 97, n. 2, p. 115–137, 1949. Available: <<http://dx.doi.org/10.2307/1538291>>. Citation on page 141.

CHAN, Y. S.; NG, H. T. Estimating class priors in domain adaptation for word sense disambiguation. In: **Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2006. (ACL-44), p. 89–96. Available: <<http://dx.doi.org/10.3115/1220175.1220187>>. Citation on page 99.

CHAPMANN, R. **The Insects. Structure and Function**. [S.l.]: Cambridge University Press, Cambridge, 2004. Citation on page 27.

CHEN, Y.; WHY, A.; BATISTA, G. E.; MAFRA-NETO, A.; KEOGH, E. Flying Insect Classification with Inexpensive Sensors. **J Insect Behav**, v. 27, n. 5, p. 657–677, 2014. ISSN 08927553. Citations on pages 31, 36, and 52.

CORTEZ, P.; CERDEIRA, A.; ALMEIDA, F.; MATOS, T.; REIS, J. Modeling wine preferences by data mining from physicochemical properties. **Decision Support Systems**, Elsevier, v. 47, n. 4, p. 547–553, 2009. Citation on page 71.

DHEERU, D.; TANISKIDOU, E. K. **UCI Machine Learning Repository**. 2017. Available: <<http://archive.ics.uci.edu/ml>>. Citations on pages 68, 87, 111, and 128.

DIETTERICH, T. G.; KONG, E. B. **Machine learning bias, statistical bias, and statistical variance of decision tree algorithms**. [S.l.], 1995. Citations on pages 24 and 120.

DYER, K. B.; CAPO, R.; POLIKAR, R. Compose: A semisupervised learning framework for initially labeled nonstationary streaming data. **IEEE Trans Neural Netw Learn Syst**, v. 25, n. 1, p. 12–26, 2014. ISSN 2162-237X. Citations on pages 35, 38, and 48.

ESULI, A.; SEBASTIANI, F.; ABBASI, A. Sentiment quantification. **IEEE Intelligent Systems**, IEEE Computer Society, v. 25, n. 4, p. 72–79, 2010. Citation on page 78.

FANAEE-T, H.; GAMA, J. Event labeling combining ensemble detectors and background knowledge. **Prog Artif Intell**, v. 2, n. 2, p. 113–127, 2014. ISSN 2192-6360. Available: <<http://dx.doi.org/10.1007/s13748-013-0040-3>>. Citations on pages 35 and 48.

FAWCETT, T.; FLACH, P. A. A response to webb and ting's on the application of roc analysis to predict classification performance under varying class distributions. **Machine Learning**, Springer, v. 58, n. 1, p. 33–38, 2005. Citation on page 28.

FORMAN, G. Counting positives accurately despite inaccurate classification. In: **SPRINGER. European Conference on Machine Learning**. [S.l.], 2005. p. 564–575. Citations on pages 24, 36, 37, 40, 66, 78, 80, 100, 102, 107, 122, and 123.

_____. Quantifying trends accurately despite classifier error and class imbalance. In: **ACM. Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [S.l.], 2006. (KDD '06), p. 157–166. Citations on pages 24, 66, 79, 103, 106, and 123.

_____. Quantifying counts and costs via classification. **Data Mining and Knowledge Discovery**, v. 17, n. 2, p. 164–206, 2008. ISSN 1573-756X. Available: <<http://dx.doi.org/10.1007/s10618-008-0097-y>>. Citations on pages 23, 78, and 79.

GAMA, J.; MEDAS, P.; CASTILLO, G.; RODRIGUES, P. Learning with drift detection. In: _____. **SBIA**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 286–295. ISBN 978-3-540-28645-5. Citations on pages 38 and 64.

GAO, W.; SEBASTIANI, F. From classification to quantification in tweet sentiment analysis. **Social Network Analysis and Mining**, Springer Vienna, v. 6, n. 1, 2016. ISSN 18695469. Available: <<http://dx.doi.org/10.1007/s13278-016-0327-z>>. Citations on pages 26, 37, 39, 40, 41, 79, 102, and 123.

GENOUD, A. P.; BASISTYY, R.; WILLIAMS, G. M.; THOMAS, B. P. Optical remote sensing for monitoring flying mosquitoes, gender identification and discussion on species identification. **Applied Physics B**, Springer, v. 124, n. 3, p. 46, 2018. Citation on page 141.

GONZÁLEZ-CASTRO, V.; ALAIZ-RODRÍGUEZ, R.; ALEGRE, E. Class distribution estimation based on the hellinger distance. **Information Sciences**, v. 218, p. 146 – 164, 2013. ISSN 0020-0255. Citations on pages 24, 25, 29, 32, 66, 78, 79, 80, 99, 105, 123, 124, and 152.

GONZÁLEZ, P.; CASTAÑO, A.; CHAWLA, N. V.; COZ, J. J. D. A review on quantification learning. **ACM Computing Surveys (CSUR)**, ACM, v. 50, n. 5, p. 74, 2017. Citations on pages 29, 66, 78, 79, 99, and 122.

GONZÁLEZ, P.; DÍEZ, J.; CHAWLA, N.; COZ, J. J. del. Why is quantification an interesting learning problem? **Progress in Artificial Intelligence**, Springer, v. 6, n. 1, p. 53–58, 2017. Citations on pages 23, 26, 79, 96, and 119.

GUEDES, D. R.; PAIVA, M. H.; DONATO, M. M.; BARBOSA, P. P.; KROKOVSKY, L.; ROCHA, S. W. d. S.; SARAIVA, K. L.; CRESPO, M. M.; REZENDE, T. M.; WALLAU, G. L. *et al.* Zika virus replication in the mosquito culex quinquefasciatus in brazil. **Emerging microbes & infections**, Taylor & Francis, v. 6, n. 1, p. 1–11, 2017. Citation on page 140.

HAMMAMI, N.; BEDDA, M. Improved tree model for arabic speech recognition. In: IEEE. **ICCSIT**. Chengdu, China, 2010. v. 5, p. 521–526. Citations on pages 48 and 71.

HILL, D. J.; MINSKER, B. S. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. **Environmental Modelling & Software**, Elsevier, v. 25, n. 9, p. 1014–1022, 2010. Citation on page 35.

HOPKINS, D. J.; KING, G. A method of automated nonparametric content analysis for social science. **American Journal of Political Science**, Wiley Online Library, v. 54, n. 1, p. 229–247, 2010. Citation on page 23.

KAHN, M. C.; CELESTIN, W.; OFFENHAUSER, W. *et al.* Recording of sounds produced by certain disease-carrying mosquitoes. **Science (Washington)**, American Association for the Advancement of Science, p. 335–6, 1945. Citation on page 26.

KAWAKUBO, H.; PLESSIS, M. C. D.; SUGIYAMA, M. Computationally efficient class-prior estimation under class balance change using energy distance. **IEICE TRANSACTIONS on Information and Systems**, The Institute of Electronics, Information and Communication Engineers, v. 99, n. 1, p. 176–186, 2016. Citation on page 125.

KIFER, D.; BEN-DAVID, S.; GEHRKE, J. Detecting change in data streams. In: VLDB ENDOWMENT. **Proceedings of the Thirtieth international conference on Very large data bases-Volume 30**. [S.l.], 2004. p. 180–191. Citations on pages 32, 62, 65, and 72.

KOLMOGOROV, A. Sulla determinazione empirica di una legge di distribuzione. **Inst. Ital. Attuari, Giorn.**, v. 4, p. 83–91, 1933. Citation on page 84.

KORU, A. G.; ZHANG, D.; LIU, H. Modeling the effect of size on defect proneness for open-source software. In: IEEE. **Predictor Models in Software Engineering, 2007. PROMISE'07: ICSE Workshops 2007. International Workshop on**. [S.l.], 2007. p. 10–10. Citations on pages 87 and 111.

KUNCHEVA, L. I. *et al.* Nearest neighbour classifiers for streaming data with delayed labelling. In: IEEE. **ICDM**. [S.l.], 2008. p. 869–874. Citation on page 64.

LEMKE, C.; BUDKA, M.; GABRYS, B. Metalearning: a survey of trends and technologies. **Artificial intelligence review**, Springer, v. 44, n. 1, p. 117–130, 2015. Citation on page 110.

LICHMAN, M. **UCI Machine Learning Repository**. 2013. <<http://archive.ics.uci.edu/ml>>. Citation on page 71.

LINCOFF, G. H. **The audubon society field guide to North American mushrooms**. [S.l.], 1989. Citation on page 87.

LYON, R. J.; STAPPERS, B.; COOPER, S.; BROOKE, J.; KNOWLES, J. Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. **Monthly Notices of the Royal Astronomical Society**, Oxford University Press, v. 459, n. 1, p. 1104–1123, 2016. Citations on pages 87 and 111.

MALETZKE, A.; HASSAN, W.; REIS, D. dos; BATISTA, G. The importance of the test set size in quantification assessment. In: **Submitted to the SIAM International Conference on Data Mining**. [S.l.: s.n.], 2020. Citation on page 33.

MALETZKE, A.; MILARÉ, C.; NADAI, B.; SAROLI, J.; SINGH, S.; CORBI, J.; MAFRANETO, A.; KEOGH, E.; BATISTA, G. Automatic insect recognition with optical sensors with variability of temperature and humidity. In: **American Mosquito Control Association 84th Annual Meeting**. [S.l.: s.n.], 2018. Citation on page 69.

MALETZKE, A.; REIS, D.; BATISTA, G. Quantification in data streams: Initial results. In: **BRACIS**. Uberlândia: [s.n.], 2017. p. 43–48. Citations on pages 31, 37, 41, 45, 49, 50, 65, and 151.

MALETZKE, A.; REIS, D. dos; BATISTA, G. **Combining instance selection and self-training to improve data stream quantification, online supplementary material**. 2018. Available: <<https://sites.google.com/site/andregustavom/research/sqsi-is>>. Citation on page 49.

MALETZKE, A.; REIS, D. dos; CHERMAN, E.; BATISTA, G. On the need of class ratio insensitive drift tests for data streams. In: **Proceedings of the Second International Workshop on Learning with Imbalanced Domains: Theory and Applications**. ECML-PKDD, Dublin, Ireland: PMLR, 2018. (Proceedings of Machine Learning Research, v. 94), p. 110–124. Available: <<http://proceedings.mlr.press/v94/maletzke18a.html>>. Citations on pages 81 and 152.

MALETZKE, A.; REIS, D. dos; CHERMAN, E.; BATISTA, G. E. A. P. A. Dys: a framework for mixture models in quantification. In: **Proceedings of the The Thirty-Third AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2019. (AAAI '19). Citations on pages 32, 106, 107, 108, 110, 113, 114, 121, 124, 126, and 129.

MALETZKE, A. G.; FERRERO, C. A.; TIBES, C. M.; CHERMAN, E. A.; ZALEWSKI, W. Medical time series classification using global and local feature extraction strategies. **Journal of Health Informatics**, v. 9, n. 3, 2017. Citation on page 150.

MALETZKE, A. G.; HASSAN, W.; REIS, D. M. dos; BATISTA, G. E. Accurately quantifying under score variability. In: **Submitted to the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining**. San Diego: [s.n.], 2020. p. ?? Citation on page 34.

MALETZKE, A. G.; REIS, D. M. dos; BATISTA, G. E. Combining instance selection and self-training to improve data stream quantification. **Journal of the Brazilian Computer Society**, SpringerOpen, v. 24, n. 1, p. 12, 2018. Citations on pages 30, 32, and 151.

MASUD, M.; GAO, J.; KHAN, L.; HAN, J.; THURASINGHAM, B. M. Classification and novel class detection in concept-drifting data streams under time constraints. **IEEE Trans Knowl Data Eng**, v. 23, n. 6, p. 859–874, 2011. ISSN 1041-4347. Citation on page 38.

MASUD, M. M.; GAO, J.; KHAN, L.; HAN, J.; THURASINGHAM, B. Classification and novel class detection in concept-drifting data streams under time constraints. **TKDE**, IEEE, v. 23, n. 6, p. 859–874, 2011. Citation on page 64.

MELLANBY, K. Humidity and insect metabolism. **Nature**, v. 138, p. 124–125, 1936. Citation on page 48.

MILLI, L.; MONREALE, A.; ROSSETTI, G.; GIANNOTTI, F.; PEDRESCHI, D.; SEBASTIANI, F. Quantification trees. In: **ICDM**. Dallas: [s.n.], 2013. p. 528–536. ISBN 978-0-7695-5108-1. Citations on pages 37 and 39.

_____. Quantification trees. In: IEEE. **2013 IEEE 13th International Conference on Data Mining**. [S.l.], 2013. p. 528–536. Citations on pages 104, 107, and 114.

MORO, S.; CORTEZ, P.; RITA, P. A data-driven approach to predict the success of bank telemarketing. **Decision Support Systems**, Elsevier, v. 62, p. 22–31, 2014. Citation on page 87.

NAEINI, M. P.; COOPER, G. F. Binary classifier calibration using an ensemble of near isotonic regression models. In: IEEE. **2016 IEEE 16th International Conference on Data Mining (ICDM)**. [S.l.], 2016. p. 360–369. Citations on pages 111 and 129.

NGUYEN, H.-L.; WOON, Y.-K.; NG, W.-K. A survey on data stream clustering and classification. **KAIS**, Springer London, v. 45, p. 535–569, 2015. ISSN 0219-1377. Available: <<http://dx.doi.org/10.1007/s10115-014-0808-1>>. Citations on pages 35, 38, and 64.

PÉREZ-GÁLLEGO, P.; CASTAÑO, A.; QUEVEDO, J. R.; COZ, J. J. del. Dynamic ensemble selection for quantification tasks. **Information Fusion**, Elsevier, v. 45, p. 1–15, 2019. Citations on pages 80, 81, and 107.

PEZZIN, A.; SY, V.; PUGGIOLI, A.; VERONESI, R.; CARRIERI, M.; MACCAGNANI, B.; BELLINI, R. Comparative study on the effectiveness of different mosquito traps in arbovirus surveillance with a focus on wnv detection. **Acta tropica**, Elsevier, v. 153, p. 93–100, 2016. Citation on page 143.

POLLARD, D. **A user's guide to measure theoretic probability**. [S.l.]: Cambridge University Press, 2002. Citations on pages 66 and 81.

REIS, D. dos; MALETZKE, A.; BATISTA, G. Unsupervised context switch for classification tasks on data streams with recurrent concepts. In: **ACM/SIGAPP**. Pau, France: ACM, 2018. v. 33. Citations on pages 48, 70, 72, and 135.

REIS, D. dos; MALETZKE, A.; SILVA, D. F.; BATISTA, G. E. A. P. A. Classifying and counting with recurrent contexts. In: ACM. **Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [S.l.], 2018. (KDD '18), p. 1983–1992. ISBN 978-1-4503-5552-0. Citations on pages 28, 32, 62, 64, 65, 67, 70, 78, 80, 81, 87, 105, 111, 112, 135, 151, and 152.

REIS, D. M. dos; FLACH, P.; MATWIN, S.; BATISTA, G. E. Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. In: **ACM SIGKDD**. San Francisco: [s.n.], 2016. p. 1545–1554. ISBN 9781450342322. Citations on pages 32, 38, 45, 58, 62, 65, and 72.

REIS, D. M. dos; MALETZKE, A. G.; CHERMAN, E.; BATISTA, G. E. One-class quantification. In: **Proceedings of the European Conference on Machine Learning**. Dublin: [s.n.], 2018. p. 564–575. Citations on pages 67, 80, 96, 101, 102, 107, and 135.

RUBNER, Y.; TOMASI, C.; GUIBAS, L. J. A metric for distributions with applications to image databases. In: IEEE. **Computer Vision, 1998. Sixth International Conference on**. [S.l.], 1998. p. 59–66. Citation on page 83.

SAERENS, M.; LATINNE, P.; DECAESTECKER, C. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. **Neural computation**, MIT Press, v. 14, n. 1, p. 21–41, 2002. Citation on page 103.

SALIM, Z. T.; HASHIM, U.; ARSHAD, M. M.; FAKHRI, M. A.; SALIM, E. T. Frequency-based detection of female aedes mosquito using surface acoustic wave technology: Early prevention of dengue fever. **Microelectronic Engineering**, Elsevier, v. 179, p. 83–90, 2017. Citations on pages 27 and 141.

SANTOS, D. A.; RODRIGUES, J. J.; FURTADO, V.; SALEEM, K.; KOROTAEV, V. Automated electronic approaches for detecting disease vectors mosquitoes through the wing-beat frequency. **Journal of Cleaner Production**, Elsevier, v. 217, p. 767–775, 2019. Citation on page 143.

SCHAFFNER, F.; KAUFMANN, C.; HEGGLIN, D.; MATHIS, A. The invasive mosquito aedes japonicus in central europe. **Medical and veterinary entomology**, Wiley Online Library, v. 23, n. 4, p. 448–451, 2009. Citation on page 137.

SCHOLTE, E.-J.; DIJKSTRA, E.; BLOK, H.; VRIES, A. D.; TAKKEN, W.; HOFHUIS, A.; KOOPMANS, M.; BOER, A. D.; REUSKEN, C. Accidental importation of the mosquito aedes albopictus into the netherlands: a survey of mosquito distribution and the presence of dengue virus. **Medical and veterinary entomology**, Wiley Online Library, v. 22, n. 4, p. 352–358, 2008. Citation on page 137.

SEBASTIANI, F. Evaluation measures for quantification: An axiomatic approach. **arXiv preprint arXiv:1809.01991**, 2018. Citations on pages 87, 111, and 129.

SETTLES, B. Active learning literature survey. **University of Wisconsin, Madison**, v. 52, n. 55-66, p. 11, 2010. Citations on pages 42 and 51.

SHIRABAD, J. S.; MENZIES, T. **The PROMISE Repository of Software Engineering Databases**. 2005. School of Information Technology and Engineering, University of Ottawa, Canada. Available: <<http://promise.site.uottawa.ca/SERepository>>. Citation on page 87.

SILVA, D. F.; SOUZA, V. M.; ELLIS, D. P.; KEOGH, E. J.; BATISTA, G. E. Exploring low cost laser sensors to identify flying insect species. **Journal of Intelligent & Robotic Systems**, Springer, v. 80, n. 1, p. 313–330, 2015. Citations on pages 23, 31, and 119.

SILVA, D. F.; SOUZA, V. M. A.; ELLIS, D.; KEOGH, E.; BATISTA, G. E. Exploring low cost laser sensors to identify flying insect species. **J Intell Robot Syst**, v. 80, n. 1, p. 313–330, 2015. ISSN 1573-0409. Available: <<http://dx.doi.org/10.1007/s10846-014-0168-9>>. Citations on pages 36 and 62.

SILVA, T. P. da; SOUZA, V. M. A.; BATISTA, G. E. A. P. A.; CAMARGO, H. de A. A fuzzy classifier for data streams with infinitely delayed labels. In: SPRINGER. **Iberoamerican Congress on Pattern Recognition**. [S.l.], 2018. p. 287–295. Citation on page 151.

SOUZA-NETO, J. A.; POWELL, J. R.; BONIZZONI, M. Aedes aegypti vector competence studies: A review. **Infection, Genetics and Evolution**, Elsevier, 2018. Citation on page 137.

SOUZA, V.; REIS, D.; MALETZKE, A.; BATISTA, G. Challenges in benchmarking stream learning algorithms with real-world data. **Submitted to Data Mining and Knowledge Discovery**, ??, n. ??, p. 1–59, 2019. Citations on pages 136, 137, 138, 139, 143, 144, and 145.

SOUZA, V. M.; ROSSI, R. G.; BATISTA, G. E.; REZENDE, S. O. Unsupervised active learning techniques for labeling training sets: An experimental evaluation on sequential data. **Intelligent Data Analysis**, v. 21, n. 5, p. 1061–1095, 2017. ISSN 1088467X. Citations on pages 42, 43, 49, and 51.

SOUZA, V. M. A.; SILVA, D. F.; GAMA, J.; BATISTA, G. E. Data Stream Classification Guided by Clustering on Nonstationary Environments and Extreme Verification Latency. In: **SDM**. Vancouver: [s.n.], 2015. p. 873–881. Citation on page 38.

STEFANO, C. D.; MANIACI, M.; FONTANELLA, F.; FRECA, A. S. di. Reliable writer identification in medieval manuscripts through page layout features: The “avila” bible case. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 72, p. 99–110, 2018. Citation on page [128](#).

STORKEY, A. When training and test sets are different: characterizing learning transfer. **Dataset shift in machine learning**, Citeseer, p. 3–28, 2009. Citation on page [28](#).

TASCHE, D. Exact fit of simple finite mixture models. **Journal of Risk and Financial Management**, Multidisciplinary Digital Publishing Institute, v. 7, n. 4, p. 150–164, 2014. Citations on pages [103](#) and [123](#).

_____. Does quantification without adjustments work? **arXiv preprint arXiv:1602.08780**, 2016. Citation on page [102](#).

_____. Confidence intervals for class prevalences under prior probability shift. **arXiv preprint arXiv:1906.04119**, 2019. Citations on pages [34](#), [120](#), and [125](#).

TAYLOR, L. Analysis of the effect of temperature on insects in flight. **Journal of Animal Ecology**, v. 32, n. 1, p. 99–117, 1963. Citation on page [48](#).

TIBES, C. M.; CHERMAN, E. A.; MALETZKE, A. G.; ZEM-MASCARENHAS, S. H.; MARTINS, J. C. A.; EVORA, Y. D. M. Software protótipo para mensurar classificação e área de lesões por pressão. In: **Simpósio APTFeridas**. [S.l.: s.n.], 2017. Citation on page [149](#).

VANSCHOREN, J.; RIJN, J. N. van; BISCHL, B.; TORGO, L. Openml: Networked science in machine learning. **ACM SIGKDD Explorations Newsletter**, ACM, New York, NY, USA, v. 15, n. 2, p. 49–60, 2013. Citations on pages [68](#), [87](#), [111](#), and [128](#).

VAPNIK, V. **The nature of statistical learning theory**. [S.l.]: Springer science & business media, 2013. Citation on page [96](#).

VILLARREAL, S. M.; WINOKUR, O.; HARRINGTON, L. The impact of temperature and body size on fundamental flight tone variation in the mosquito vector aedes aegypti (diptera: Culicidae): Implications for acoustic lures. **Journal of medical entomology**, Oxford University Press, v. 54, n. 5, p. 1116–1121, 2017. Citations on pages [27](#), [136](#), and [141](#).

VUK, M.; CURK, T. Roc curve, lift chart and calibration plot. **Metodoloski zvezki**, Anuska Ferligoj, v. 3, n. 1, p. 89, 2006. Citations on pages [24](#) and [37](#).

WISHART, G.; RIORDAN, D. Flight responses to various sounds by adult males of aedes aegypti (l.)(diptera: Culicidae). **The Canadian Entomologist**, Cambridge University Press, v. 91, n. 3, p. 181–191, 1959. Citation on page [26](#).

WU, X.; LI, P.; HU, X. Learning from concept drifting data streams with unlabeled data. **Neurocomputing**, Elsevier, v. 92, p. 145–155, 2012. Citation on page [64](#).

YEH, I.-C.; LIEN, C.-h. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. **Expert Systems with Applications**, Elsevier, v. 36, n. 2, p. 2473–2480, 2009. Citation on page [87](#).

ZHU, X.; GOLDBERG, A. B. Introduction to semi-supervised learning. **Synthesis lectures on artificial intelligence and machine learning**, Morgan & Claypool Publishers, v. 3, n. 1, p. 1–130, 2009. Citations on pages [43](#) and [49](#).

ŽLIOBAITĖ, I. Change with delayed labeling: when is it detectable? In: IEEE. **ICDMW**. [S.l.], 2010. p. 843–850. Citation on page 72.

ŽLIOBAITĖ, I. Learning under concept drift: an overview. **Computing Research Repository**, abs/1010.4784, 2010. Available: <<http://arxiv.org/abs/1010.4784>>. Citation on page 65.

ZLIOBAITE, I.; BIFET, A.; PFAHRINGER, B.; HOLMES, G. Active learning with drifting streaming data. **IEEE transactions on neural networks and learning systems**, IEEE, v. 25, n. 1, p. 27–39, 2014. Citations on pages 42 and 43.