UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Non-stationary and unpredictable data distributions in classification and quantification

Denis Moreira dos Reis

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: _____

Denis Moreira dos Reis

Non-stationary and unpredictable data distributions in classification and quantification

Thesis submitted to the Institute of Mathematics and
Computer Sciences – ICMC-USP – in accordance with
the requirements of the Computer and Mathematical
Sciences Graduate Program, for the degree of
Doctor in Science. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Prof. Dr. Gustavo Enrique de Almeida Prado
Alves Batista

Co-advisor: Profa. Dra. Elaine Parros Machado
de Sousa

USP – São Carlos
June 2020

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

d375n dos Reis, Denis Moreira
Non-stationary and unpredictable data
distributions in classification and quantification
/ Denis Moreira dos Reis; orientador Gustavo
Enrique de Almeida Prado Alves Batista;
coorientadora Elaine Parros Machado de Sousa. --
São Carlos, 2020.
175 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, 2020.

1. dados não estacionários. 2. dados positivos e
não-rotulados. 3. quantificação. 4. classificação. 5.
mudança de conceito. I. Enrique de Almeida Prado
Alves Batista, Gustavo, orient. II. Parros Machado
de Sousa, Elaine, coorient. III. Título.

Denis Moreira dos Reis

**Distribuições não estacionárias e imprevisíveis em
classificação e quantificação**

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. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Gustavo Enrique de Almeida Prado Alves Batista

Coorientadora: Profa. Dra. Elaine Parros Machado de Sousa

**USP – São Carlos
Junho de 2020**

*This work is dedicated to all people
who are unable to dedicate something.*

ACKNOWLEDGEMENTS

I could not open this manuscript without thanking my friends and family members for all support I've received. They are many and growing by the day. Special thanks go to my parents and sister, who were always present and supportive when I really needed.

I heartfully thank my beloved Rita, who kept me awake from my daydreaming.

I also thank my advisor Prof. Batista, for whom, at times, I have been a heavier burden than he could have anticipated. He did not drop the weight, and for that he has my gratitude.

Once Prof. Batista moved overseas, Prof. Parros welcomed me with open arms, and I cannot thank her enough for personally supporting my final months in this journey.

Finally, I'm immensely thankful that I could collaborate with André Maletzke, who rocketed many of my ideas off the ground.

Additional thanks go to FAPESP and CAPES, that financially supported this thesis (grants 2017/22896-7 and PROEX-6909543/D, respectively).

*“‘All conditioned things are impermanent’
— when one sees this with wisdom,
one turns away from suffering.”
(The Buddha)*

RESUMO

DOS REIS, D. M. **Distribuições não estacionárias e imprevisíveis em classificação e quantificação.** 2020. 175 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, 2020.

Nos últimos anos, é crescente a preocupação da academia com dados não estacionários. Por um lado, diferenças entre as distribuições dos dados disponíveis em laboratório, para treino e avaliação, e dos dados encontrados posteriormente, após implantação dos modelos para uso prático na aplicação, levam ao decréscimo de performance em diversas tarefas, como classificação. Por outro, em outras tarefas, como quantificação, há o interesse explícito em mensurar como a distribuição dos dados se altera ao longo do tempo. Em quaisquer casos, porém, é comum a existência de premissas que tornam as soluções disponíveis ao mesmo tempo possíveis em seus campos de estudo, e impráticas para aplicações reais. Esta tese de doutorado propõe soluções com premissas menos restritivas e/ou mais realistas, que possibilitem seu emprego em aplicações reais. Na área de detecção de mudança de conceito, são introduzidas técnicas de detecção não supervisionada que possibilitam classificação e quantificação sem a necessidade de requisitar rótulos verdadeiros após implantação do modelo. Em quantificação, é iniciada uma nova sub-área de pesquisa, *one-class quantification*. De forma similar à *one-class classification*, em *one-class quantification* não é assumida uma distribuição particular para a classe negativa, sendo considerada, portanto, *imprevisível*. Os resultados obtidos pela avaliação experimental empregada são promissores e demonstram a viabilidade de alternativas mais focadas em solucionar os problemas existentes em aplicações do mundo real.

Palavras-chave: dados não estacionários, aprendizado com dados positivos e não-rotulados, quantificação, classificação, mudança de conceito.

ABSTRACT

DOS REIS, D. M. **Non-stationary and unpredictable data distributions in classification and quantification**. 2020. 175 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, 2020.

In the last years, we observed a crescent academic interest on nonstationary data. On the one hand, differences between the data that was used to induce a model and the data that is found after the model is deployed cause a decrease of performance for several tasks, such as classification. On the other hand, in several tasks, such as quantification, we are explicitly interested in measuring how a distribution changes over time. For any of these problems, however, we generally run into solutions that rely on strong assumptions, which are impractical or even impracticable in real world applications. In this thesis, we provide solutions that rely on less restrictive and/or more realistic assumptions in order to allow such methods to be employed in real applications. In the concept drift detection area, we introduce unsupervised drift detection methods that allow for performing classification and quantification without ever requesting true labels after deployment. In the quantification area, we bootstrap a new research topic called *one-class quantification*. Similarly to *one-class classification*, in *one-class quantification* we are able to avoid strong assumptions regarding the negative class, which is deemed unpredictable. Our experimental results are promising and provide favorable evidences about the viability of solutions that are focused on solving real-world problems.

Keywords: nonstationary data, positive and unlabeled learning, quantification, classification, concept drift.

LIST OF FIGURES

Figure 1 – Schema of intelligent sensor	38
Figure 2 – Influence of temperature on wing-beat frequency	38
Figure 3 – Circadian rhythm of mosquitoes	39
Figure 4 – Correlation between WBF and temperature	41
Figure 5 – Synthetic data distribution	42
Figure 6 – Distribution of collected data according to altitude	43
Figure 7 – Distribution of collected data according to temperature	43
Figure 8 – Circadian rhythm of <i>Ae aegypti</i> and <i>Ae albopictus</i>	44
Figure 9 – Distribution of collected data according to WBF	46
Figure 10 – Histogram of insects data across varying temperature	47
Figure 11 – Histogram of insects data across varying temperature and humidity	47
Figure 12 – Density of collected data for <i>An quadrimaculatus</i>	49
Figure 13 – Illustration of class overlap	52
Figure 14 – Practical difference between materialization and observational sequence	55
Figure 15 – Illustration of a mixture of distributions plausibly caused by concept drift	57
Figure 16 – Patterns of change in a data stream	58
Figure 17 – Circadian rhythm of <i>Culex quinquefasciatus</i>	59
Figure 18 – Illustration of temporal overlap	61
Figure 19 – Temporal overlap in real-world data	62
Figure 20 – Undetectable concept drift	65
Figure 21 – Case of traceable drift	65
Figure 22 – Examples of practical changes in data	67
Figure 23 – Illustration of density function of scores for classification and balanced classes	71
Figure 24 – Match between theoretical and empirical performance of Classify and Count	72
Figure 25 – Illustration of density function of scores for classification and unbalanced classes	73
Figure 26 – Effect of sub-classes to a scorer	75
Figure 27 – Effect of unknown sub-class to a scorer	76
Figure 28 – Illustration of general case of Positive and Unlabeled Learning problem	77
Figure 29 – HDy scheme	88

Figure 30 – Densities for the contexts in the WBF-Insects data set	90
Figure 31 – Kolmogorov-Smirnov statistic over time for the WBF-Insects data stream	91
Figure 32 – XO-HDy schematics	95
Figure 33 – Context identification accuracy for all positive class proportions for dataset Aedes-Culex	102
Figure 34 – Comparison between adjusted and non-adjusted classification perfor- mance across different class proportions for XO-HDy on Aedes-Culex .	104
Figure 35 – ORD versus other dissimilarity functions	106
Figure 36 – Search space of DySyn	110
Figure 37 – Relation between density of MoSS scores and uniform distribution . . .	111
Figure 38 – ROC curve for MoSS	112
Figure 39 – TPR and FPR of MoSS	112
Figure 40 – Elkan’s algorithm	118
Figure 41 – Illustration of the <i>knee</i> finding process in AlphaMax	121
Figure 42 – Mahalanobis distance	126
Figure 43 – One-class SVM	127
Figure 44 – Local Outlier Factor	128
Figure 45 – Isolation Forest	129
Figure 46 – Rationale behind PAT	132
Figure 47 – MAE for different values of q in PAT	133
Figure 48 – Rationale behind ODIn	134
Figure 49 – Histograms in ODIn	134
Figure 50 – Nemenyi test for Experiment #1	143
Figure 51 – Nemenyi test for Experiment #3-a	144
Figure 52 – Nemenyi test for Experiments #3-b and #3-c	145
Figure 53 – One-class Naive Bayes	148
Figure 54 – Situation where TIcE performs better than PAT	152

LIST OF ALGORITHMS

Algorithm 1 – Ordinal Distance	106
Algorithm 2 – Sample ORD	107
Algorithm 3 – TIcE	123
Algorithm 4 – ExTIcE	136

LIST OF TABLES

Table 1 – Correlation between environment and WBF	48
Table 2 – Tabular data representation	52
Table 3 – WBF overlap between female <i>Ae. aegypti</i> and <i>Cx quinquefasciatus</i> for varying temperature	62
Table 4 – Example of perfect quantification with imperfect classifier	70
Table 5 – Number of events for <i>WBF-Insects</i> dataset	89
Table 6 – Ternary search suitability for the HDy algorithm	101
Table 7 – Context identification accuracy for SMR and XO-HDy	101
Table 8 – HDy absolute quantification error averaged over all positive proportions	102
Table 9 – ACC absolute quantification error averaged over all positive proportions	103
Table 10 – Classification accuracy averaged for all positive class proportions	103
Table 11 – Mean classification accuracy for all positive class proportions with ad- justed decision threshold	104
Table 12 – Comparison of classification accuracy between MINAS (FARIA <i>et al.</i> , 2016b) and XO-HDy in data stream setting	105
Table 13 – Context identification accuracy for SMR-SORD and XO-SORD	108
Table 14 – Experiment #1 regarding OCQ and PU PE	143
Table 15 – Experiment #3-a regarding OCQ and PU PE	144
Table 16 – Experiment #3-b regarding OCQ and PU PE	145
Table 17 – Experiment #3-c regarding OCQ and PU PE	146
Table 18 – Experiment #4 regarding OCQ and PU PE	147
Table 19 – AUC of one-class scorers for insects	149
Table 20 – Experimental RanFoCE	150
Table 21 – Experimental results for ensemble with PAT and ExtIcE	153
Table 22 – Empirical comparison between PAT_M and $ODIn_M$	175
Table 23 – Experiment #2 regarding OCQ and PU PE	175

LIST OF ABBREVIATIONS AND ACRONYMS

CC	Classify and Count.
FPR	False Positive Ratio.
HD	Hellinger Distance.
IF	Isolation Forest.
LOF	Local Outlier Factor.
MD	Mahalanobis Distance.
MUDA	Multi-source Domain Adaptation.
OCQ	One-class Quantification.
OCS	One-class Scorer.
ONB	One-class Naive Bayes.
OSVM	One-class SVM.
PU L	Positive and Unlabeled Learning.
PU PE	Positive and Unlabeled Prior Estimation.
SUDA	Single-source Domain Adaptation.
TPR	True Positive Ratio.
WBF	Wing-beat frequency.

LIST OF SYMBOLS

$\mathcal{X}$ — Feature space.

$\mathcal{Y}$ — Set containing all possible class-labels.

x — A data point.

$y(x)$ — Actual class-label of x .

D_X — Data points in set D disregarding their class-labels.

X — Random variable that represents one observation point.

Y — Random variable that represents the class-label of one observation point.

$P(X,Y)$ — Distribution of data.

$P_A(X,Y)$ — Distribution of data under context A .

L — Labeled dataset.

$h(x)$ — Classifier.

U — Unlabeled dataset

t_h — Classification threshold.

$h^s(x)$ — Scorer.

$h^v(x,c)$ — Class-confidence for data point x about class c .

$q(D)$ — Quantifier.

p — Positive ratio, that is, actual proportion of data points that belong to the positive class.

$\hat{p}$ — Estimation of p .

$P_h(\oplus|\oplus)$ — True positive ratio of classifier h .

$P_h(\oplus|\ominus)$ — False positive ratio of classifier h .

$P_h(\ominus|\ominus)$ — True negative ratio of classifier h .

$P_h(\ominus|\oplus)$ — False negative ratio of classifier h .

$s(x)$ — A function that denotes whether data point x is labeled (in which case, $s(x) = 1$).

c — In the context of PU L and OCQ, $c = P(s(\mathbf{x}) = 1|y(\mathbf{x}) = c_+)$.

$\mathcal{C}$ — Set of contexts.

CONTENTS

1	INTRODUCTION	29
1.1	Hypothesis and contributions	34
1.2	Organization	34
2	PRACTICAL MOTIVATION	37
2.1	Problem description	37
2.2	Biases in data collection	40
2.3	Outcomes and possible solutions	45
2.4	Final considerations	49
3	CONCEPTS	51
3.1	Changes in data	51
3.1.1	<i>Data representation and organization</i>	51
3.1.2	<i>Batch learning versus stream learning</i>	53
3.1.3	<i>Patterns of change in data streams</i>	56
3.1.4	<i>Framing distributions</i>	60
3.1.5	<i>Types of change</i>	62
3.2	Machine learning tasks	67
3.2.1	<i>Classification</i>	67
3.2.2	<i>Scoring</i>	68
3.2.3	<i>Quantification</i>	69
3.2.4	<i>Open-set Quantification</i>	73
3.2.5	<i>Positive and Unlabeled Prior Estimation</i>	75
3.3	Final considerations	78
4	CONTEXT IDENTIFICATION	81
4.1	Related work	81
4.1.1	<i>Label latency</i>	82
4.1.2	<i>Unsupervised drift detection</i>	84
4.1.3	<i>Domain adaptation</i>	86
4.1.4	<i>Binary quantification</i>	87
4.2	Practical motivation	89
4.3	Our proposals	93
4.3.1	<i>SMR-HDy</i>	93

4.3.2	<i>XO-HDy</i>	94
4.3.3	<i>Classification adjustment</i>	95
4.3.4	<i>Computational requirements</i>	95
4.4	Experimental setup	97
4.4.1	<i>Algorithms and Evaluation</i>	97
4.4.2	<i>Datasets Description</i>	99
4.5	Experimental evaluation	100
4.5.1	<i>Quantification</i>	101
4.5.2	<i>Classification</i>	103
4.5.3	<i>Data Stream Classification</i>	104
4.6	Related additional contributions	105
4.6.1	<i>Sample ORD</i>	105
4.6.2	<i>Mixable Kolmogorov-Smirnov</i>	108
4.6.3	<i>Model for Score Simulation</i>	109
4.7	Discussion	113
5	ONE-CLASS QUANTIFICATION	115
5.1	Related work	117
5.1.1	<i>Positive and Unlabeled Prior Estimation</i>	117
5.1.1.1	<i>Elkan</i>	117
5.1.1.2	<i>PE and pen-L1</i>	119
5.1.1.3	<i>AlphaMax</i>	119
5.1.1.4	<i>KM</i>	121
5.1.1.5	<i>Tree Induction for c Estimation (Tlce)</i>	121
5.1.2	<i>One-class scorers</i>	125
5.1.2.1	<i>Mahalanobis Distance (MD)</i>	125
5.1.2.2	<i>One-class SVM (OSVM)</i>	126
5.1.2.3	<i>Local Outlier Factor</i>	127
5.1.2.4	<i>Isolation Forest</i>	129
5.2	Our proposals	129
5.2.1	<i>Differences between OCQ and PU PE</i>	130
5.2.2	<i>Passive-Aggressive Threshold</i>	131
5.2.3	<i>One Distribution Inside</i>	133
5.2.4	<i>Exhaustive Tlce</i>	136
5.3	Experimental setup	136
5.3.1	<i>Experiment #1</i>	137
5.3.2	<i>Experiment #2</i>	137
5.3.3	<i>Experiment #3</i>	138
5.3.4	<i>Experiment #4</i>	138
5.3.5	<i>Algorithms</i>	139

5.3.6	<i>Datasets</i>	140
5.4	Experimental evaluation	142
5.5	Related additional contributions	147
5.5.1	<i>One-class Naive Bayes</i>	147
5.5.2	<i>Random Forest for C Estimation</i>	149
5.6	Discussion	151
6	CONCLUSION	155
BIBLIOGRAPHY		161
GLOSSARY		171
APPENDIX A ONE-CLASS QUANTIFICATION		173
A.1	Analysis on TlcE's time complexity	173
A.2	Comparison between PAT and ODIn	174
A.3	Exp #2	174

INTRODUCTION

Literature on Data Mining has a long history of assuming stationarity of data. Simply put and broadly speaking, according to this assumption, data collected in controlled environments to feed Machine Learning algorithms, in order to train computational models (training data), follows the same statistical distribution as data that is later observed when such models go into effect (deployment data).

The stationarity assumption, which holds for several real-world applications, enabled scholars and practitioners alike to progress in all sorts of tasks, such as classification, clustering and, equally important, evaluation methods to fairly compare all methods proposed along the way.

However, a method is only useful as long as its requirements are fulfilled. Overlooking changes in the probabilities of data from training to deployment may lead to severe performance loss. As previously demonstrated in the literature, classification systems trained with a certain distribution heavily misclassify when deployment data follows a different distribution (PAN; YANG, 2009; QUIONERO-CANDELA *et al.*, 2009; SAENKO *et al.*, 2010; BEN-DAVID *et al.*, 2007). Indeed, for a broad range of applications, such as the prediction of electricity supply and demand (HARRIES, 1999; ZHU, 2010) or the prediction of ozone levels (DHEERU; TANISKIDOU, 2017), expecting the distribution of the data to be immutable is plainly unreasonable.

Generally, the mismatch between training data and deployment data occurs according to the passage of time or a change of the physical location where data is observed. One familiar example of data that changes over time is voice, since one's voice changes drastically after their puberty. Applications such as voice recognition systems are impacted by such alterations. One example where data changes depending on where it is acquired is the set of answers obtained through a questionnaire about politics. A model induced using the answers from one country will hardly represent the answers from another

country. Such mismatches are known as several names such as concept/dataset/covariate drift/shift. When each data point has an associated class-label (among a limited set of known classes), the easiest type of change to deal with arguably is the *prior probability shift* (MORENO-TORRES *et al.*, 2012). In this type of change, the only mismatch between training and deployment data is their differing proportions of data points that belong to each class.

In this regard, a new and rapidly evolving area of research, named Quantification (GONZÁLEZ *et al.*, 2017), is interested in detecting and quantifying changes in the proportions of the classes, while assuming the nonexistence of any other change.

In several applications, we may be interested in counting the number of occurrences of a specific event – our class of interest. For instance, we may be interested in counting the number of times a specific species of frog croack in the field (DIAZ *et al.*, 2012; LICHMAN, 2013) while disregarding other species and other animals altogether. Although we may have enough data to fairly characterize the class of interest, that is, the expected event, it may be impractical to collect enough data to fully characterize everything that *is not* expected.

In this sense, an unexplored problem, yet present in many real-world applications, is our proposed One-class Quantification. Particularly, the term one-class refers to the interest in estimating the number of individuals of one specific class. This class of interest is called the *positive class*, which can be modeled with training data. Everything else falls into the name of *negative class*, which will have presumably unpredictable behavior after deployment. In this work, we understand as unpredictable distribution one that cannot be fairly modeled with training data.

To the best of our knowledge, the only area of research that focuses on this problem is Positive and Unlabeled Prior Estimation (ELKAN; NOTO, 2008; PLESSIS; SUGIYAMA, 2014; JAIN *et al.*, 2016; RAMASWAMY; SCOTT; TEWARI, 2016; BEKKER; DAVIS, 2018). However, our experience with those methods indicates they present inferior results both in terms of quantification performance and computational costs when compared to our proposals.

Putting prior probability shift aside, other types of change are also frequently present in typical machine learning applications. A considerable portion of recent papers that tackle such changes of distribution have focused their efforts on a specific setting referred to as *data streams*. In data stream applications, we are usually interested in continuously observing and analysing a series of data points (called data stream) over time (FEIGENBAUM *et al.*, 2002). In this context, the extraction of knowledge from a stream is referred to as Data Stream Mining (GAMA *et al.*, 2010).

Stream mining is *usually* associated with real time applications, where data points

must be analyzed as soon as they are observed and the system must be ready to analyze the next observation in the sequence before it is made available. Due to a stream’s potentially infinite length and the fast pace with which data points are observed, data streams often impose severe memory and processing time constraints (BIFET, 2009).

For instance, it may be expected that the distribution of data points change *while* the stream is still being observed (WIDMER; KUBAT, 1996). For this reason, differently from batch mining, there is often no hard boundaries between different distributions: the stream can be a continuous observation of an evolving distribution.

Generally speaking, we can expect two types of streams. In the first, observations are made at regular intervals, as is the case with temperature sensors, stock prices, etc. In the second type, the availability of data can vary over time. Hence, a period of scarcity of data can obscure the perception of the distribution’s evolution.

In the last decade, there has been a tremendous increase of interest in methods that can learn from data streams and overcome their related problems. As a consequence, numerous algorithms for different tasks, such as classification (NGUYEN; WOON; NG, 2015; GOMES *et al.*, 2017a; KRAWCZYK *et al.*, 2017), clustering (SILVA *et al.*, 2013; GUHA; MISHRA, 2016), and novelty and anomaly detection (HILL; MINSKER, 2010; FARIA *et al.*, 2016a) have been proposed.

A particular task of interest in data stream mining is classification, where we want to infer the class-label associated with unlabeled data points. To tackle the possible change of distribution, a significant portion of literature assumes that true labels are instantaneously available as soon as the classifier issues a prediction (GAMA *et al.*, 2004; BIFET; HOLMES; PFAHRINGER, 2010; OZA, 2005; BIFET; GAVALDA, 2007a; BIFET; GAVALDA, 2009). Real-world applications hardly conform with such an assumption. If the true labels are available instantly, there is no need for predicting them in the first place. However, to obtain true labels at some point in time while still observing the stream is plausible in certain applications. One example is the task of predicting whether stock prices are going up or down one hour ahead of time. We can observe all true labels with a difference of exactly one hour from their respective predictions and, during this time, other predictions are being made.

Nevertheless, for the majority of real-world problems, the availability of true labels is due to factors that are out of the machine learning practitioner’s control. Additionally, obtaining true labels may involve a variety of costs, such as hiring domain specialists for manual classification. There is also an inherent time interval that is needed to obtain the true label. For instance, the time required for the domain specialist to manually classify an event.

Instantaneous availability of *all* true labels provides an advantageous and usually

unrealistic position to the classification algorithms, enabling them to detect concept drifts and adapt faster than they actually could in practice. This fact is more prominently observed in detection methods that depend on the analysis of classification performance over time.

For benchmark classification tasks, the results reported in the body of work that assumes instantaneous availability of labels are overly optimistic. One could argue that this fact is not a real issue, since the interest with the experiments is to derive the relative performance of classifiers, that is, to identify which algorithm is the best for a given task, regardless of the measured performance value. However, this objective implies the assumption that changing label availability does not change the relative performance of different classifiers, which is false (REIS, 2016).

On the other end of the spectrum, there are problems where we cannot obtain true labels while the stream is still being observed. The lack of this information not only complicate the drift detection in general, but also turns it impossible to detect some types of change (ŽLIOBAITĖ, 2010). While one can argue that the types of change that are undetectable seem unnatural to most real-world data, they are nonetheless present in synthetic data sets that are frequently used in experimental evaluations found in literature, as the hyperplane generator in the highly employed MOA framework (BIFET *et al.*, 2010).

Previous work has been successful at building tools to detect changes in distribution without relying on labels (KIFER; BEN-DAVID; GEHRKE, 2004; ŽLIOBAITĖ, 2010; REIS *et al.*, 2016; KORYCKI; KRAWCZYK, 2019). However, they fail to provide solutions to the underlying problem (classification) after a change of distribution is detected. Without access to true labels, it may not be possible to adequately induce a new or adapt the existing classification model.

Another approach to the lack of true labels is to track the evolution of the distribution as it changes (DYER; CAPO; POLIKAR, 2014; SOUZA *et al.*, 2015; FARIA *et al.*, 2016b). Methods that follow this approach are restricted to incremental changes and are prone to accumulate error over exceedingly long streams of data. Although it is possible to assume that only incremental changes can happen in certain applications, we emphasize two important aggravating factors. First, periods of time in which data is unavailable can obscure the evolution of the distribution. In practice, an incremental change can present itself as a sudden and severe change, depending on how much data was actually observed by the system. Second, these methods require training data to comprise data points at the beginning of the stream. In reality, training data is usually collected under differing circumstances, such as in a laboratory. We show a practical example that presents both issues in the next chapter.

A third approach is to consider several training distributions and relate at least one of them to the distribution found in deployment data. In this regard, literature provides

methods under the task known as Domain Adaptation (KUMAGAI; IWATA, 2017; XU *et al.*, 2018; ZHAO *et al.*, 2018; ZHU; ZHUANG; WANG, 2019). In this setting, the relation between deployment data and training data is usually ill-defined, as well as the motivating factor for different distributions to begin with. Perhaps for this very reason, domain adaptation methods suffer from three main issues. One, they generally require a vast amount of data for the deep learning approaches they rely on. Second, for each set of unlabeled data observed after deployment, the methods require considerable computational resources to infer the relations between all distributions. Third, and more importantly, they generally require a large feature space that can be reduced to a subspace where data is invariant across all distributions.

To the best of our knowledge, no previous work has tackled the possibility of unsupervisedly reusing previously induced models to directly solve the classification task at hand. This represents an untapped opportunity to enable a fully unsupervised approach for data mining with concept drifts.

To summarize, we approach changing data in two distinct topics: *(a)* one-class quantification, in which we provide new methods that improve the state-of-the-art in terms of smaller quantification error and computational requirements; *(b)* context identification, in which we propose methods that cover both classification and quantification in settings where the distribution of deployment data changes.

Regarding context identification, we answer the following questions:

1. Is it possible to automatically and unsupervisedly switch between different models that better represent the current data, given that such data share similarities with some previously analyzed data?
2. Can the switch mechanism work under abrupt changes of distribution?
3. Can the switch process disregard prior probability shifts that are unlikely to affect the classification performance?
4. Is it better to keep several models and switch to one among them rather than using a single classification model induced over the whole available data?

For the one-class aspect of changing data, we answer the following questions over the course of this thesis:

- Contrary to methods of Positive and Unlabeled Prior Estimation, can we induce a one-class quantification that can be reused to quantify several samples of data, without needing to be trained again?

- Can we obtain smaller quantification error and smaller computation requirements than the state-of-the-art?

With such questions raised, we define our hypothesis and list our contributions in the next section.

1.1 Hypothesis and contributions

In this thesis, we raise the following hypotheses:

1. Given an *unlabeled* data set U and a set of classifiers $\mathcal{H}$, it is possible to identify which classifier $h \in \mathcal{H}$ best models the likelihoods ($P(X|Y)$) found in U ; and
2. Quantification methods specifically designed for one-class scenarios can outperform the state-of-the-art in terms of quantification error and computation cost.

By confirming the given hypothesis, our main contributions are:

1. The development of methods that can efficiently and accurately identify which classification model, among those in a given set, best classifies current *unlabeled* data;
2. The development of methods that explicitly differentiate and measure prior probability shift from other changes, while still identifying those changes that can impact classification performance; and
3. The development of quantification methods that tackle one-class settings.

1.2 Organization

In [Chapter 2](#), we expose a real-world problem that motivates this work and in which we face all discussed challenges regarding data drift. We additionally elaborate on possible data collection bias issues that directed our research efforts.

In [Chapter 3](#), we extensively discuss the main concepts that are employed throughout this work, such as which types of data we are interested in, how they can be represented and, more importantly, the changes that they are susceptible to. We end the chapter overviewing the base tasks that make use of such data and how they are affected by their non-stationarity.

In this thesis, we cover two main topics that are reasonably distinct: context identification, discussed in [Chapter 4](#), and one-class quantification, discussed in [Chapter 5](#). To provide a more concise reading, we make a clear separation between the topics. For each

one, we provide a brief introduction followed by literature review, our contributions and conclusions on the topic.

Finally, in [Chapter 6](#), we summarize our contributions, relate them to our motivating application, emphasize their limitations, and discuss prospects for future research.

PRACTICAL MOTIVATION

In this chapter, we discuss a real-world application in which we face the main challenges posed by concept drift. We first open this chapter by describing the problem, how it is affected by different types of changes, and its social importance. In the following section, we analyze how biases in the data collection process can poison our efforts by leading to unrealistically good performance expectations.

2.1 Problem description

In the last years, our research group has designed a sensor for flying insects ([SILVA *et al.*, 2015](#)). This particular sensor captures data regarding the flight of insects that pass through a plane formed by aligned infrared LEDs and phototransistors. [Figure 1](#) illustrates the sensor. The registered data is later processed by machine learning algorithms to classify the insect signals into species and sex.

This sensor can augment existing mosquito traps, creating a device that can attract, capture, and also count the insects. Hence, such trap can be a valuable tool for mosquito control and surveillance.

For control, classification is the primary task. Real-time classification allows us to create a trap that only captures insects of interest, such as disease vectors or agricultural pests. Other species can be released, reducing the environmental impact of the device.

For surveillance, quantification is the primary task. Counting the number of captured insects that belong to a species and sex of interest gives us an estimate of the insect population in the trap area. Heat maps integrate the counts of multiple traps and provide an essential tool for planing and executing larger-scale control interventions.

Data collected by the trap present drifts. There are a large set of factors that influence the flying behavior of insects such as temperature, humidity, air pressure, age,

Figure 1 – Schema of intelligent sensor.

Source: Batista *et al.* (2011).

availability of water and food and so forth. Among those, temperature is a particularly prominent factor. Figure 2 illustrates temperature influence on the wing-beat frequency, one of the features we extract from the signals.

Figure 2 – Influence of temperature on the wing-beat frequency of *Aedes aegypti* mosquitoes.

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

An aggravating factor is that the class distribution is unknown and highly variable. It depends on two primary factors: the local availability of the species of interest and the insect circadian rhythm. The circadian rhythm is a biological process that governs peaks of activity and resting. For many insect species, these peaks occur at dawn and dusk, as shown in Figure 3.

For this application, we possess plenty of labeled *training* data for a limited number

Figure 3 – Histograms representing the circadian rhythm of *Aedes aegypti* and *Culex quinquefasciatus* mosquitoes.

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

of species. We gather these data in the laboratory using insectaries, which are containers with a sensor attached. Each insectary maintains insects of a single species, leading to labeled data. We also control temperature, humidity, and other ambient conditions maintaining the insectaries in climatized chambers.

However, training data offers limited information regarding class distribution. In laboratory, the class distribution depends on the number of insects in each insectary and the circadian rhythm of the insects. However, on the field, the class distribution will be given by the local availability of the insects as well as their circadian rhythms.

Additionally, laboratory conditions differ from those in the deployment field. As such, we cannot count on the same data availability. In laboratory, some factors such as true labels are known only because we directly control them. Those promptly become unavailable after deployment.

While we can measure many of the factors that influence the insects' behavior through sensors, we would want to remove all non-essential ones, since they raise battery consumption, building costs, and project complexity. The ambient sensors are particularly troublesome since they must be installed externally to better register the ambient conditions. Therefore, they are also prone to damages, resulting in failures.

Nevertheless, the possibility of controlling some factors (such as temperature, humidity, air pressure, and so on) that affect the wing beat data enables us to collect enough data to identify a reasonable portion of all recurrent behavioral patterns.

We note that other applications may fall into one of four similar situations that follow:

1. The factors responsible for different data distributions are unknown, too many, or

too complex to be feasibly accountable. In this case, it may still be possible to recognize distinct and recurrent contexts over time (by applying a suitable statistical test, for instance);

2. Even if the factors responsible for drifts are known, they can be missing in data obtained with a different setup;
3. The sensors responsible for registering data related to the drifts are prone to failures. Hence a fallback mechanism is desired; and
4. The sensors are unreliable. Hence the use of their readings can be potentially improved with an additional opinion, as in a data fusion scheme.

To summarize, data collected in a laboratory: *(a)* misrepresents the prior probability of different species and; *(b)* does not include all possible species that can be observed in the field since this exhaustive enumeration is impossible. In this research, we tackle both issues, and advance two new and growing areas of research: unsupervised drift detection and one-class quantification.

In the next section, we further detail the technical difficulties regarding data collection for this particular application. This case study directed and justified choices made during this research project.

2.2 Biases in data collection

In this section, we conduct an analysis of the data available for the insect sensor problem. Our findings from this study guided our requirements and proposals in this work.

The data was collected as described in the previous section. To that description, we add that data were gathered by multiple sensors across multiple regions of the globe, under diverse environmental conditions. However, the diversity of circumstances the sensors underwent is not shared by all classes: certain species were collected under some environmental conditions, but not under every possible condition.

When an insect passes through the field-of-view of a sensor, it registers a sequence containing levels of light obstruction. The final result is a time series similar to what would have been obtained by recording the audio with a microphone, with the benefit of the former method being exempt from external audible interference, such as bird sounds and noise from passing cars (BATISTA *et al.*, 2011). From the time series, we extract several features, where wing-beat frequency (WBF) is the singular most distinctive of them. In this analysis, we only consider WBF. Additional sensors register the following environmental conditions: temperature, humidity, luminosity, altitude, and air pressure.

As previously discussed, our data indicates considerable correlation between environmental factors, particularly temperature, and flight characteristics of insects. The correlation between temperature and WBF of female *Aedes aegypti* is illustrated in Figure 4.

Figure 4 – Correlation between wing-beat frequency and temperature.

Source: Research data.

As determining the temperature clearly segregates the expected behavior of this species, it is a valuable information that can improve the performance of models that seek to identify the species of insects. Ideally, we would provide a Machine Learning algorithm with the temperature, in addition to the characteristics of flight, and the algorithm would make the best use of this information to induce a model. However, as we will see in this section, doing so incurs unrealistic and severely optimistic evaluation of the resulting model, which in turn supports bad decisions.

We start our discussion with a toy example. Suppose a problem where we have two classes, c_1 and c_2 , and we are tasked to create a model that can classify unlabeled data points: that is, the model must tell whether a data point belongs to c_1 or c_2 . Both classes are a mixture of two one-dimensional Gaussian distributions, where the first is centered at one and the second is centered at three. First, let us suppose that all our data regarding this problem is distributed according to the density function illustrated in Figure 5a and that our data is balanced, that is, we have an equal amount of data points for each class. In this case, if both our training and test sets are non-intersecting and uniform samples from the same data (as is the case with k -fold cross validation), we would achieve an accuracy of around 50%. Indeed, as both classes behave exactly the same way (according to our descriptive attribute), we cannot tell the difference between the two.

Suppose now that *all* our data is distributed according to the density function illustrated in Figure 5b, and we once again have balanced data, meaning a same number of data points for each class. In this new scenario, although both classes are mixtures

Figure 5 – Synthetic data distribution.

Source: Elaborated by the author.

of Gaussian distributions that share the same parameters, the proportion of data points that are generated by each Gaussian varies from class to class. The best possible classifier trained with this data would guess that whenever a data point is below two, it belongs to c_2 . Conversely, if it is above two, the best guess is that it belongs to c_1 . Although it does not perfectly classify every data point, this classifier would achieve an accuracy significantly greater than 50%, given that both test and training sets are uniform samples from our available data.

The objective of a Machine Learning algorithm is to create a model that maximizes some evaluation metric. In order to maximize accuracy (and any other usual measure related to classification), the model will learn local proportion of classes. More specifically, it will learn to classify data points that fall into a region with the predominant class of said region, *according to the training data*.

Figure 5a and Figure 5b have a significant difference in how distinguishable c_1 is from c_2 . This raises questions: which one is representative of the real-world? Is it possible that Figure 5a represents reality while Figure 5b is skewed due to some bias in the data collection process? Is there a true and stationary distribution, or it may vary even in the real-world? All those questions relate with the insect sensor problem.

Insect data for different species were collected across several days, under different environmental conditions, seasons of the year, locations, and with varying number of insectaries containing varying number of specimens. For some environmental variables, this process resulted in a clearly skewed distribution. Figure 6 shows the distribution of data according to the altitude where they were collected, for female *Aedes aegypti* and female *Aedes albopictus*. We can clearly see that if a Machine Learning algorithm is fed with this data, it will incorrectly learn that anything that has an altitude below 400 units is *Aedes aegypti*. In reality, this data only relates to the physical location where the collection of the data took place.

Figure 6 – Distribution of collected data for female *Aedes aegypti* and female *Aedes albopictus* according to the altitude where they were physically collected.

Source: Research data.

Consider the less extreme situation of temperature, displayed in Figure 7a. For the sake of clarity, consider only two temperature ranges: from 23°C to 25°C, and from 31°C to 33°C. The distribution of data when only data points within such ranges are considered is shown in Figure 7b. We can see that we have considerably more data for female *Aedes aegypti* within [31°C, 33°C] than we have within [23°C, 25°C]. Meanwhile, the amount of data we have for *Aedes albopictus* does not diverge as much from one range to the other. We note that the plots in Figure 7 are smoothed distributions that were estimated empirically and, due to characteristics of the estimation algorithm, Figure 7b is not a perfect slice of Figure 7a, although the former was estimated with a subsample of the same data.

Figure 7 – Distribution of collected data for female *Aedes aegypti* and female *Aedes albopictus* according to the ambient temperature where they were physically collected.

(a) All temperatures.

(b) Only within two ranges.

Source: Research data.

Let us assume that the data is balanced: the total number of *Aedes aegypti* is the same as the number of *Aedes albopictus*. With this available data, we can see that insects observed under temperatures above 28°C are more likely to be classified as female *Aedes*

aegypti, even before looking at the flight characteristics. This fact may seem odd, since temperature by itself is not a part of the insect. And yet, temperature influences the probability of the output of the classifier (which is a species). This influence would not be a problem by itself if all proportions were credible. After all, it may be the case that temperature affects the activity of insects the same way the circadian rhythm does.

Figure 8 – Circadian rhythm of *Aedes aegypti* and *Aedes albopictus*.

Source: Research data.

Figure 8 presents the circadian rhythm of both species under study. We can see that the activity of the species vary differently over the course of a day. Particularly, *Aedes albopictus* considerably decreases its activity between 5 p.m. and 5 a.m. Considering this figure alone, we cannot say that an insect observed at 3 a.m. is more likely to be an *Aedes aegypti* than *Aedes albopictus*. These distributions depict different *intra-class* proportions.

Each bin of the histogram represents the chance of observing a mosquito at a given time in relation to the chance of observing a mosquito of the *same species*, but at a different time. We cannot say which species is more likely to be observed at any given time since we do not have the true proportion of the classes, that is, how many *Aedes aegypti* exist more (or less) than *Aedes albopictus*. Regarding *our data*, the proportion of the classes relates to how many insectaries we have, and how many specimens each insectary contains. Those are values we decide and unlikely represent what happens in natural habitats, where mosquitoes live and reproduce. For instance, suppose that we have the same number of insectaries with the same number of insects of both species. Suppose that, at a given time, *Aedes albopictus* is half as active as *Aedes aegypti*. Our model would favor classifying an insect as *Aedes aegypti*, in this situation. However, if in the field there actually are twice as many *Aedes albopictus* as *Aedes aegypti*, the chance of observing either is the same. Factually, we expect the proportion of mosquitoes to vary in the field according to season, precipitation levels, presence of predators, geographical location and so on. Therefore, the proportion of insects in the field is rather unpredictable.

However, in the case of Figure 7b, in addition to the misrepresentation of the proportion of one class in relation to the other, we also have a misrepresentation of one

class in relation to itself for varying values of temperature. Contrary to circadian rhythm, the distribution of data according to temperature depends on the number of times we collected data and the experimental setup of each time. Such variables vary arbitrarily according to how we proceeded with data collection, and do not reflect reality.

In the next section, we discuss the effects of the misrepresented data and possible ways to circumvent this problem.

2.3 Outcomes and possible solutions

Our initial expectation was that, by providing environmental information to a classifier, it would better use flight information to classify insects. However, given the misrepresentation, we actually risk biasing the classifier. One possible outcome is making it learn how to model the conditions in which data was collected, since identifying experimental setup strongly relates to our skewed proportion of classes. Another possible outcome is making the classifier fail to consider some environmental conditions. For example, if we try to model the data points with Gaussian functions, points that were obtained in less frequent experimental settings might be mistakenly considered noise or part of the tail of another well established Gaussian rather than its own. By not taking our data's misrepresentation into account, evaluation methods like k -fold cross validation can provide unrealistic and overly optimistic performance levels for our attempts at classifying data, in contrast to how the same classification models would perform in reality. The problem lies on drawing test data points from the same biased source distribution from which we draw training data points.

The conclusion is *not* that we should simply give up on environmental information, for two reasons. First, by doing so we lose information that can be useful to decrease the overlap between classes, as explained in [Section 3.1.4](#). Second, and more importantly, by simply ditching environmental information we ensure that it is impossible to remove bias in the future.

Our initial assumption is that environmental variables correlate with flight characteristics. Therefore, skewed conditions during collection incur skewed flight characteristics. [Figure 9a](#) presents the distribution of WBF obtained with the data that is distributed according to [Figure 7b](#). If we balance the data so that the number of observation points within each temperature range is the same, the distribution of WBF changes to what is shown in [Figure 9b](#). Notice how *Aedes aegypti* and *Aedes albopictus* behave visibly more similar (according to WBF) in the latter case.

We have to consider the proportion of classes for each environmental condition. We emphasize that balancing the classes so that they have the same number of data points will not make our data reflect reality. Such a change only replaces one bias with another

Figure 9 – Distribution of collected data for female *Aedes aegypti* and female *Aedes albopictus* according to their wing-beat frequency at the moment they were observed.

Source: Research data.

one. In this application, the proportion of the classes *will* be biased and *will not* represent the real-world regardless of the proportion we chose, for two reasons: first, we cannot assess the real proportion by the way we collect data; and second, the real proportion is non-stationary and vary over time and space. Therefore, in this application, we are always making a decision on *how* the bias is, consciously or otherwise. With that in mind, we argue that, for classification, balancing the classes is the best decision: the classifier will learn to expect that all classes have a same chance of being observed within each environmental condition, and will have to look for flight characteristics that can really tell the classes apart. Similarly, we can induce separate models for each condition to avoid misrepresenting intra-class proportions.

We note that balancing data is not trivial. For instance, data collected for *Aedes aegypti* massively outnumbers *Aedes albopictus*, as illustrated in Figure 10. Taking only temperature into account, undersampling the former class incurs losing tens of thousands of data points. Conversely, oversampling the latter incurs the replication of a few points tens of thousands of times. Literature about unbalanced data offers other methods to tackle this particular problem. In any case, the unbalance problem is due to the class with the fewest data points.

In the previous examples, we considered temperature alone. Including additional environmental variables increases the need for data exponentially, since each environmental condition is described by a combination of all variables considered. Aside from requiring more data so that we have enough for each combination, we also decrease the chance of having conditions where enough data for all classes were collected. This becomes ever more problematic as we add more classes. Figure 11 compares the distribution of data across both temperature and humidity for *Aedes aegypti* and *Aedes albopictus*.

Giving our limited data, we may have to choose a limited number of environmental variables to represent an environmental condition. In Table 1, we measure the Spearman

Figure 10 – Histogram counting the number of data points observed across different temperature ranges, for female *Aedes aegypti* and female *Aedes albopictus*.

Source: Research data.

Figure 11 – Histogram counting the relative number of data points observed across different combinations of temperature and humidity ranges, for female *Aedes aegypti* and female *Aedes albopictus* individually.

Source: Research data.

correlation between each environmental variable and WBF for each class we have available.

Table 1 – Spearman correlation between each environmental feature and wing-beat frequency. The biggest absolute value of each row is bolded.

class	temperature	humidity	luminosity	altitude	air pressure
♀ <i>Ae aegypti</i>	0.49	0.06	-0.02	-0.09	0.09
♂ <i>Ae aegypti</i>	0.77	-0.46	-0.01	0.12	-0.12
♀ <i>Ae albopictus</i>	0.80	-0.31	-0.11	0.30	-0.30
♂ <i>Ae albopictus</i>	0.64	-0.42	-0.09	0.36	-0.36
♀ <i>An aquasalis</i>	0.53	0.30	-0.17	-0.07	0.07
♂ <i>An aquasalis</i>	0.59	-0.20	-0.03	0.02	-0.02
♀ <i>An quadrimaculatus</i>	0.29	0.06	-0.31	0.01	-0.01
♂ <i>An quadrimaculatus</i>	0.26	0.02	-0.30	0.17	-0.17
♂ <i>Chironomos</i>	0.25	0.01	-0.04	0.07	-0.07
♀ <i>Cx quinquefasciatus</i>	0.48	-0.16	0.00	0.29	-0.29
♂ <i>Cx quinquefasciatus</i>	0.49	-0.08	-0.14	0.12	-0.12
♀ <i>Cx tarsalis</i>	0.43	-0.13	0.00	0.38	-0.38
♂ <i>Cx tarsalis</i>	-0.14	-0.23	-0.16	-0.16	0.16

Source: Research data.

Unsurprisingly, temperature consistently presents the biggest correlation among all variables considered. One notable exception is *Anopheles quadrimaculatus*, that has its WBF *apparently* more correlated with luminosity. The odd aspect of this result is the lack of correlation between luminosity and WBF for other classes. We note that this finding can be explained by the sparsity of conditions in which data was collected along with malfunctioning sensor during data collection.

Figure 12 illustrates the behavior of female *Anopheles quadrimaculatus* regarding luminosity and WBF, as a pair, and of luminosity and temperature, as another pair. In Figure 12a, observe that there are two distinguishable clusters of data points. The clusters are only separated by a wide interval of luminosity for which we severely lack data. Each cluster contains roughly half of all data points, and they present different distributions for both WBF and temperature (hence the high absolute Spearman correlation). However, within each cluster, we have small values of absolute correlation (0.02 and -0.04). Finally, the cluster on the right has abnormally high numbers of luminosity in comparison to other classes. The average luminosity for female and male *Anopheles quadrimaculatus* are 525 and 397 units, respectively. The class with the third largest average luminosity is male *Aedes aegypti*, at 25 units.

We note that solely misrepresenting the proportion of classes in relation to each other is not a problem if our task is quantification. In this case, such a misrepresentation is *expected* as part of the problem. However, misrepresenting the distribution of data *within* a class leads to a number of problems for quantification algorithms: since it skews our performance evaluation for classification, methods that rely on that information such

Figure 12 – Density of data for pairs of environmental variables. Data collected from female *Anopheles quadrimaculatus*.

Source: Research data.

as ACC (FORMAN, 2005) – see Section 3.2.3 – cannot operate properly; methods that model the distribution of data, such as AlphaMax (JAIN *et al.*, 2016) – see Section 5.1.1.3 –, are not provided with reliable representation of the distribution.

2.4 Final considerations

To summarize our findings, we should avoid directly providing environmental variables to machine learning algorithms since otherwise we can end up learning experimental setup instead of insect species, or inducing a model of a class that potentially ignores or smooths out points collected under underrepresented environmental conditions. We can separate the data into chunks of distinct environmental conditions and induce models for each chunk as a possible approach for making use of environmental data. We note that misrepresenting the proportion of classes in training data is not an additional problem for quantification, since such a misrepresentation is already expected, and that quantification can help classification achieve better performance (REIS *et al.*, 2018b). Finally, given the volume of data required to properly model each environmental condition, we should favor one-class approaches so that we can focus our collection efforts on only a handful of species.

In the next chapter, we better formalize the changes that can occur in data and their implications for several tasks. After that, in the following chapters, we review and introduce new methods that can be applied to tackle the problems discussed in this

section.

CONCEPTS

The contents of this chapter are based on our contributions to a published conference paper (REIS *et al.*, 2018a) and to a journal paper that was recently submitted and is under review (SOUZA *et al.*, 2020).

In this chapter, we explain concepts that are needed throughout the remaining of this thesis. This chapter is split into two main parts. In the first of them, we formalize how data is represented and organized, and proceed to examine the different ways by which its distribution can change. In the second part, we formalize and discuss different tasks that depend on data and are directly affected by changes in data distribution: scoring, classification, quantification, and positive and unlabeled learning.

3.1 Changes in data

In this section, we discuss the different settings in which data can change, and how those changes take place. Before moving forward with this topic, however, we need to formalize how data is represented and discuss the implications of their representation.

3.1.1 Data representation and organization

In this work, we are mainly interested in structured data that adhere to the attribute-value representation (SAMMUT; WEBB, 2010): each observation point (henceforth simply called observation) can be interpreted as row in a table, where each column corresponds to one attribute. Each observation is also associated with a class label, *i.e.*, it has a nominal class attribute. All other attributes, excepting the class attribute, constitute the *feature space* $\mathcal{X}$. Table 2 illustrates how data can be represented as a table.

Formally, a data set $D = \{(x_1, y(x_1)), \dots, (x_n, y(x_n))\}$, where $x_i = \langle x_{i,1}, \dots, x_{i,m} \rangle$, $x_i \in \mathcal{X}$ and $y(x_i) \in \mathcal{Y} = \{c_1, \dots, c_l\}$, where n is the number of observation points and l is the

Table 2 – Illustration of how data can be represented as a table.

index	D_X			D_Y
	attribute 1	... attribute m		class label
1	$x_{1,1}$	... $x_{1,m}$		$y(x_1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
n	$x_{n,1}$	... $x_{n,m}$		$y(x_n)$

Source: Elaborated by the author.

number of unique classes. For completeness and to provide a matrix perspective, consider $D_X = \langle x_1, \dots, x_n \rangle$, $D_X \in \mathcal{X}^n$, and $D_Y = \langle y(x_1), \dots, y(x_n) \rangle$. In such a perspective, we can make the following simplification: $D_Y = y(D_X)$. Additionally, if all attributes from the feature space are real values, then $\mathcal{X} = \mathbb{R}^m$, where m is the number of attributes. In this case, $D_X \in \mathbb{R}^{n \times m}$.

Observe that x_i is a vector. That makes it possible for us to adopt concepts from geometry. In particular, we are interested in the concept of *class overlap*. The general meaning of class overlap is that the boundaries of a class, that is, the region of the feature space $\mathcal{X}$ in which we observe points that belong to a class, intercept the boundaries of another class. Therefore, the classes are not, at least linearly, separable. The main concern about overlap is usually how much of each class is within the overlapping region.

Figure 13 – This figure illustrates the effect of class overlap in a two dimensional feature space with two classes, c_1 and c_2 . The intersecting area contains data points from both c_1 and c_2 .

Source: Elaborated by the author.

Although we presented the general representation of a data set, there are sizable differences that occur when dealing with stream data, versus batch data. We detail such differences in the next section.

3.1.2 Batch learning versus stream learning

Although observation points in both stream learning and batch learning can be laid out as a table, the arrangement of the observations, particularly their order and the size of the table, vary significantly.

In batch learning, a data set represents a sample, with a fixed number of observations, (ideally) uniformly drawn from a stationary probability distribution. In that regard, each individual observation within this particular data set is independent and identically distributed (*i.i.d*) (UPTON; COOK, 2014). “Identically distributed” means that all observations in the set share the same underlying distribution. “Independently distributed” means that the observations are independent of each other, that is, the occurrence of one observation does not affect the probability of the occurrence of any other particular observation. Hence, the index column in Table 2 has the sole purpose of uniquely identifying observation points, and do not imply any relation between different points. In this perspective, the set notation D is suitable since the order of the data points does not matter, and a set do not impose a restrictive order to their elements.

In stream learning, a data set represents a sequence of discrete observations sampled from a usually dynamic environment. In a stream, the index value of each observation positions it within the sequence, so that there is logical order to be followed. Consider two observation points, x_i and x_j . If $i < j$, then x_i comes before x_j in the sequence. The meaning of the order, however, is not always self-evident and depends on the context of the application (SOUZA *et al.*, 2020). While the progression of the sequence is usually tied to the progression of time, this progression may relate more or less with the actual *observation* of the data point rather than their *materialization*. In this perspective, the matrix notation suits the data more than the set notation, since it imposes an order to the elements.

If the sequence is tied to the materialization of the data points, the last data point observed did not manifest in the world before its predecessor, and may even cease to exist before the observation of its successor. In this thesis, we refer to this type of sequence as “materialization sequence”.

An example that follows this perspective is the analysis of time-series, as it is the case of analysing prices or ambient sensors over time (HARRIES, 1999; DHEERU; TANISKIDOU, 2017; ZHU, 2010). We note that, typically, the observable data is complete, although it must be discretized. In other words, all data points that come to existence can be observed, at the moment of their materialization. In such cases, each data point is a function of the previous one. In that sense, the evolution of the data can be tracked by simpler methods, such as measuring the difference between consecutive points. The practical objective is generally to predict future readings or the trend of the data. In that

regard, individual readings are of little importance.

On the other hand, if a sequence is tied to the actual observation rather than the manifestation, then the order implies only that x_i was observed (by the data collection procedure) before x_j , even though x_j may have existed in a moment prior to the materialization of x_i . In this work, we refer to this type of sequence as “observational sequence”.

Among several possible examples (ZLIOBAITE, 2011; IKONOMOVSKA; GAMA; DVEROSKI, 2011; KATAKIS *et al.*, 2009; SOUZA *et al.*, 2015b; REIS *et al.*, 2016), we emphasize the application of intelligent sensors for insects. Multiple insects of diverse species can coexist in the vicinity of the sensor at the same time. Meanwhile, the sensor will only observe and register flight data of insects that fly right in front of it. Therefore, the resulting sequence relates to the order of flights that happened in front of the sensor, and disregards the actual order in which insects appeared in the region. In cases that follow this structure, the observable data contains only a fraction of all data that exist: in the example, not all insects might fly in front of the sensor. In such cases, events are independent of each other, in the sense that observing one particular data point does not influence the next particular data point. All data points will nevertheless manifest according to a background and shared probability distribution, that may evolve over time. For instance, insect data is affected by the ambient temperature. If the practical objective is to take autonomous action for each observation point (trap or blow out the insect depending on their characteristics), then we can say that individual readings are of great importance.

Figure 14 illustrates the practical difference between observational and materialization sequences. Granted, in real world applications, both can be combined to provide more information to a computational model. In the example, it is possible to know the temperature when the wing-beat frequency of an insect was registered.

The sequences previously discussed have a relation between their data points and time: the points are sorted according to either the time of their materialization or the time of their observation. For that reason, we refer to such sequences as “temporal sequences”.

However, not all sequences of streaming data are tied to the chronological order of events. There are cases where the sequence is tied to the spatial disposition of entities in the world. We refer to those as “spatial sequences”. One such example is the study of pavement of a highway (SOUZA; GIUSTI; BATISTA, 2018). The extent of the road is split into confined sections that are analyzed individually. The stream has a logical sequence since the order of such sections follows the path of the road in one of its directions. However, to build a data set that represents this precise stream, we are not required to collect data in the same order as it should be presented. Indeed, we can gather data from the sections in any order and sort the sequence afterwards. In this sense, time is irrelevant

Figure 14 – Plot illustrating the difference between a materialization sequence (in the form of a time-series) and an observational sequence. The blue dots and crosses are stream events. For each observation point there is an associated value of wing-beat frequency (WBF) captured by a sensor (the left y-axis presents a scale for WBF). Dots represent males and crosses represent females *Aedes aegypti* mosquitoes. The red line represents the ambient temperature (right y-axis presents a scale in Celsius). The observation of these blue events over time constitutes the observational sequence, and the temperature represents a materialization sequence.

Source: Research data.

to this stream.

Finally, we refer to sequences that follow neither the chronological order of events nor the spatial disposition of entities, and yet have a logic behind the arrangement of the observation points, as “logical sequences”.

To summarize, data streams are represented as sequences of data points. Such sequences can be grouped together into two types according to the relation between consecutive points:

materialization sequences data points are arranged in the same order as they manifest in the world; each data point heavily influences the characteristics of its successor, and is heavily influenced by the characteristics of its predecessor;

observational sequences data points are arranged in the same order as they are observed, which is not necessarily the order with which they manifested in the world; each point is distributed according to a shared background distribution, but does not directly influence the characteristics of its successor, neither is influenced by its predecessor.

In this research project, we are only interested in data sets that are *observational*

sequences, or, at least, a mixture of observational and materialization sequence where the observational part is more relevant. For this reason, this chapter is henceforth dedicated to this type of sequence. We point the interested reader to Time Series literature for a more appropriate study on materialization sequences.

Data streams can also be independently categorized according to the nature of the sequence:

temporal sequence time is the main factor that dictates the order with which data points are observed;

spatial sequence the physical disposition of data points in the world dictates the order with which they are observed;

logical sequence the data points are observed according to a logical sequence that is neither temporal or spatial;

unordered sequence data points are effectively shuffled and no two distinct distributions can be identified within the sequence.

On one hand, due to both the dynamism in the environments from which streams are sampled, and the logical disposition of events, the i.i.d. assumption usually does not hold. The dynamics of the processes that generate observation points can cause the probability distribution that feeds the stream to evolve or effectively change over the course of sequence. This means that different sub-sequences within the stream can follow significantly different probability distributions, and as such, observations are not identically distributed along the stream. Such changes, which are called concept drifts, can also bias the data towards specific class-labels, causing temporal dependence: the observation of a class changes the probability of the observation of this and/or other classes in the future. Recent studies have found a significant *temporal dependence* in real-world data (BIFET *et al.*, 2013; ŽLIOBAITĖ *et al.*, 2015). As such, observations are not independent.

On the other hand, even though we consider only one probability distribution when analyzing a data set in batch learning, we may have to apply knowledge gathered from this data set on another, different data set, that may follow a different distribution. In this sense, we also have to deal with changes in data in applications of batch learning.

In the next section, we enumerate the different patterns with which the data can change in a stream.

3.1.3 Patterns of change in data streams

To better describe how change occurs in a data stream, consider the following formalization. Let us call X the random variable that represents an observation point, and

Y a random variable that represents the corresponding class label. The joint probability $P(X,Y)$ represents the distribution of the data. A change means the transition from one distribution into another one, during the stream. In this light, consider P_A the probabilities associated with the distribution prior to the change, and P_B the probabilities associated with the distribution after the change. Consequently, $P_A(X,Y)$ is the distribution itself before change, and $P_B(X,Y)$ represents the distribution after change. Here, we only consider cases where $P_A(X,Y) \neq P_B(X,Y)$.

Naturally, if we take all data from the stream as a whole, it will likely be a sample from a mixture of *several* probability distributions. Indeed, in real-world applications, at any time, each class consists of *at least* one *distinct* distribution, otherwise it would be useless to assign class labels to data points. Having said that, we are not interested in identifying separable distributions from all data points in a stream. In fact, our objective is to identify distributions that are *temporally* separable within the stream, regardless of whether they are mixture of even inner distributions. In other words, we want to identify whether two *slices* of the stream have different distributions. A slice defines hard limits for which data points in the sequence can be considered. For instance, a slice of a temporal sequence restricts us to data points that were observed during a given interval of time. Figure 15 illustrates the distinction between identifying changes of distribution within a stream and simply identifying different distributions inside a data sample.

Figure 15 – This figure illustrates a mixture of two distributions. If the data that generates such a mixture is arranged as Figure 15b, we can consider the mixture to be caused due to changes of distribution during the stream. However, Figure 15c does not provide evidence for the same conclusion.

Source: Elaborated by the author.

Although there is an immeasurable number of ways of how a change in data can happen in a stream, those changes can still be categorized into a finite number of patterns. Gama *et al.* (2014) list several patterns for change in streams, which we further detail next.

An *abrupt change* occurs when, until a certain point in time, all data points within the sequence were distributed according to $P_A(X, Y)$ and, from that point onward, all new data points are distributed according to $P_B(X, Y)$.

To understand the *incremental change*, let us say that, until the data point x_a , all observations in the sequence are distributed according to $P_A(X, Y)$, and from x_b , all observations are distributed according to $P_B(X, Y)$. Suppose that the initial distribution $P_A(X, Y)$ is parametric with parameters $\vec{p}_A$, and the final distribution $P_B(X, Y)$ is parametric with parameters $\vec{p}_B$. The in-between observation points $x_i, a < i < b$ are distributed according to a mixture that has parameter $\alpha_i \vec{p}_A + (1 - \alpha_i) \vec{p}_B$, where $0 \leq \alpha_i \leq 1 \ \forall i$ and $\alpha_j < \alpha_k \leftrightarrow j < k$.

A *gradual change* occurs when there is a transition period from $P_A(X, Y)$ and $P_B(X, Y)$ in which the chance of observing data points that are distributed according to $P_A(X, Y)$ decreases, while the chance of observing data points distributed according to $P_B(X, Y)$ increases. Again, consider that all observations up until x_a are distributed according to $P_A(X, Y)$. From x_b , all observations are distributed according to $P_B(X, Y)$. The in-between observation points $x_i, a < i < b$ are either distributed according to $\alpha_i P_A(X, Y) + (1 - \alpha_i) P_B(X, Y)$, where $0 \leq \alpha_i \leq 1 \ \forall i$ and $\alpha_j < \alpha_k \leftrightarrow j < k$.

In addition to the three patterns of change above, two behavioral patterns of streams are noteworthy: *recurring concepts* refers to the return of a distribution that generated data points in the past but, at some point, ceased to do so; and *outliers* refers to data points that are effectively distributed according to a different distribution than other points in the stream, although this distribution does not persist in additional data points. Figure 16 illustrates all patterns discussed.

Figure 16 – Patterns of change in a data stream.

Source: Souza *et al.* (2020), Gama *et al.* (2014).

Considering the different ways a data stream is organized into a sequence, and the patterns of change, we can clearly note the importance of understanding the spacing between consecutive observation points. Although data points are *typically* observed at regular time or space intervals (in temporal or spatial sequences, consecutively), this may

not always be the case. In settings where consecutive data points within the sequence are observed at varying rates of time or space, additional complications emerge.

Take our previous example of an insect sensor for intelligent traps. Data points are observed whenever insects fly through the field of view of the sensor, and therefore at a irregular interval. As flying insects are affected by the circadian rhythm, the present peaks of activity and inactivity. [Figure 17](#) displays how active is the *Culex quinquefasciatus* species throughout a day.

Figure 17 – Density function representing the circadian rhythm of *Culex quinquefasciatus*.

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

Insects of this species become abruptly inactive at dawn, and resume their activities after dusk. During their inactive hours, even though the sensor may not be registering events and creating new observation points, time passes, environmental condition changes and, when the species starts being active again and producing new observation points, the characteristics of flight will certainly be different than before.

If we were provided with a sequence of observation points that relate exclusively to the characteristics of flight, without time stamps that indicate when each observation were made, we would identify an abrupt and inexplicable change in the distribution of the data. However, if we were also provided with time stamps, we could notice the absence of data for a prolonged period of time that would explain the otherwise inexplicable abrupt change. This example highlights the importance of temporal or spatial marks to properly understand concept drifts.

We also emphasize that similar cases of inexplicable and abrupt changes happen when moving the conditions with which we gather data from one setting to another. For example, consider that we first analyse insect data in laboratory to later resume our analysis on the field. This transition will undoubtedly impact the distribution of the data we are observing.

In the next section, we explain the need of imposing analysis windows to limit distributions within a stream. We also advocate that, generally, we can transfer the methods that are applicable in batch learning to data stream.

3.1.4 Framing distributions

In the previous section, we discussed that our main interest is in identifying distinct data distributions between slices of the stream. Such slices translate to slices of time or physical space, depending on the nature of the stream. Note that both time and space are continuous variables. Furthermore, the distribution and the drift themselves can be continuous. One example is the incremental change of temperature during a period of time. The passage of time in this period is continuous, and the temperature is continuous as well. Therefore, the background distribution that affects the flight of insects is also continuous. Nevertheless, data points, be they temperature values or the wing-beat frequency of insects, are all discrete.

Ultimately, even if the concept in a stream is continuous, we can only estimate it from samples of consecutive data points (or *observation windows*). As the number of data points is not continuous, there are a discrete number of observation windows that can be analyzed. Hence, a limited number of estimations of distributions.

If we consider only pairs of windows from the stream, we can adopt methods from batch learning and statistics to compare them, such as hypothesis tests. We note that the use of observation windows incur the selection of important parameters: how many observation points will be comprised by each window; how much is the overlap between consecutive windows, if any; and if the windows are not overlapping, how much is the spacing between them. More broadly, where each window starts and ends.

In face of the challenges of choosing such parameters, the community developed methods for change detection that allow for variable windows (HAQUE; KHAN; BARON, 2015; BIFET; GAVALDA, 2007b; HUANG *et al.*, 2014). The general idea behind those methods is that the observation windows are non-intersecting and consecutive, and should be as small as possible, given they can differentiate two distribution at a reasonable significance level.

Factually, it is possible to mimic the general intent of the aforementioned techniques with any method that compares two samples of data. However, it is a time-consuming and therefore impractical task. The peculiarity of the specific methods is their slimmer computational requirements. On the other hand, they pose limits to which methods can be adopted to actually compare windows, or provide biased samples to a third-party method. This fact hinder their applicability and may impose undesirable restrictions to certain real-world problems, such as the ones we study in this work.

We note that, regardless of the method used to choose the observation windows, smaller windows can contain too few observations to properly define the distributions they should represent. It is especially true if the distributions are a mixture of other distributions. Nonetheless, too large windows can include, within their limits, distinct temporally

separable distributions. In other words, fast-paced concept drifts can go undetected, even though they could have been detected with smaller windows.

For most applications, the negative impact of not properly identifying concept drifts lies on the creation of *temporal overlaps* (SOUZA *et al.*, 2020). Temporal overlap is a superposition of instances that belong to different classes in the feature space, and that only exists if we ignore the temporal aspect of the data. It means that if we fail to temporally split the data into slices so that we can identify distinct distributions, we incorrectly perceive a greater class overlap than otherwise. In this particular illustration, a temporal overlap could suggest that around 50% of c_1 overlaps with around 50% of c_2 in the feature space. The existence of temporal overlap reinforces the importance of adequately choosing the parameters of observation windows.

Figure 18 – Illustration of a case where there is temporal overlap. There are two classes (c_1 and c_2). Their distributions are shown before and after the drift happened. The drift is a global linear transformation that moved the average feature value two units up. The green shade illustrates a temporal overlap: instances that belong to class c_1 would seem to belong to class c_2 according to the outdated distributions.

Source: Elaborated by the author.

In Figure 19, we illustrate temporal overlap with real-world data. In this particular case, the temporal overlap is caused by a too large observation window, and can be reduced by splitting the window into smaller ones. This illustration is complemented by Table 3, which presents the numerical overlap between the two classes for each temperature. The overlap when all temperatures are considered together is 36%, while the average overlap when each temperature is isolated is 23%. The overlaps were estimated by taking the minimum between histograms with 100 bins. We use temperature as a proxy for time, since the former incrementally change as a function of the latter.

So far, we discussed the patterns which change occurs in data. In the next section, we review how such changes are.

Table 3 – Values for a case of temporal overlap. When we can discriminate the data according to the current temperature, we have smaller class overlap in the wing-beat frequency between female *Aedes aegypti* and female *Culex quinquefasciatus*.

Temperature (°C)	24	26	28	30	32	34
Overlap (%)	29	32	28	23	19	5

Source: Research data.

Figure 19 – Temporal overlap in real-world data. If we decrease the size of the observation window (originally it includes temperature from 24°C to 34°C), we also decrease the overlap between classes.

Source: Research data.

3.1.5 Types of change

In this work, since we are assigning class labels to individual observation points, we expect a correlation between the feature space $\mathcal{X}$ and the set of possible labels $\mathcal{Y}$. From this perspective, [Fawcett and Flach \(2005\)](#) state that there are two types of problems which define the causal direction of the relation between $\mathcal{X}$ and $\mathcal{Y}$. They are:

$\mathcal{X} \rightarrow \mathcal{Y}$: the class label derives from the characteristics of data point. One example is the task of associating types of postures to people: characteristics of the body determine the name of the type of posture. For problems in this category, the joint distribution is more often expressed as $P(X, Y) = P(Y|X)P(X)$;

$\mathcal{Y} \rightarrow \mathcal{X}$: the class label of a data point determines its characteristics. One example is the task of collecting flight information from insects: the species determine the characteristics of flight that are registered. For problems in this category, the joint distribution is usually expressed as $P(X, Y) = P(X|Y)P(Y)$.

Based on the types of problems described, [Moreno-Torres et al. \(2012\)](#) review and compile common nomenclatures and definitions into a list of types of changes. Although general changes are commonly referred to as *concept drift* in the data stream community,

the list provides a normalization for the nomenclature where all types of changes go under the *dataset shift* name. Subsequently, any change belongs to one among three more specific types: *covariate shift*, *prior probability shift* and *concept shift*.

Covariate shift refers to changes in the feature space $\mathcal{X}$ that do not alter how the corresponding class distribution is perceived. One example is the occurrence of changes in attributes that do not correlate with the class labels. According to [Moreno-Torres et al. \(2012\)](#), this type of shift only happens in $\mathcal{X} \rightarrow \mathcal{Y}$ problems. It is defined as follows:

Definition 1. Covariate shift is the case where $P_A(Y|X) = P_B(Y|X)$ and $P_A(X) \neq P_B(X)$.

Prior probability shift refers to changes in the proportions of the classes that do not alter how the classes imply the characteristics of data points. It is the main subject of study in a new sub-field of Machine Learning called class Prior Estimation or Quantification ([GONZÁLEZ et al., 2017](#)). According to [Moreno-Torres et al. \(2012\)](#), this type of change only happens in $\mathcal{Y} \rightarrow \mathcal{X}$ problems, and is defined as follows:

Definition 2. Prior probability shift is the case where $P_A(X|Y) = P_B(X|Y)$ and $P_A(Y) \neq P_B(Y)$.

We note that, in prior probability shift, although $P_A(X|Y) = P_B(X|Y)$, it is not necessarily true that $P_A(Y|X) = P_B(Y|X)$. This can be easily observed by changing $P(Y)$ in data sets with classes that highly overlap. To illustrate, recall that classifiers typically learn to classify data points in a region of the feature space as the most common class in said region. However, the most common class is subject to change according to alterations in $P(Y)$. Yet, the behavior of each class, individually, might remain the same.

Concept shift is a change in the relationship between the feature space and the class labels. In other words, the attributes from the feature space $\mathcal{X}$ change how they describe the different classes from $\mathcal{Y}$, and/or the classes in $\mathcal{Y}$ change the characteristics that they cause. Concept shift is defined as follows:

Definition 3. Concept shift is the case where one of the following happens:

1. $P_A(Y|X) \neq P_B(Y|X)$ and $P_A(X) = P_B(X)$ in $\mathcal{X} \rightarrow \mathcal{Y}$ problems;
2. $P_A(X|Y) \neq P_B(X|Y)$ and $P_A(Y) = P_B(Y)$ in $\mathcal{Y} \rightarrow \mathcal{X}$ problems;
3. $P_A(Y|X) \neq P_B(Y|X)$ and $P_A(X) \neq P_B(X)$ in $\mathcal{X} \rightarrow \mathcal{Y}$ problems;
4. $P_A(X|Y) \neq P_B(X|Y)$ and $P_A(Y) \neq P_B(Y)$ in $\mathcal{Y} \rightarrow \mathcal{X}$ problems.

Condition 1 states that the proportions of the classes, given the characteristics of the data points, change, while the distribution of said characteristics remains the same. In

practical terms, we have two effects. First, if we ignore the label information and compare the data before and after the drift, they have the same probability distribution. Second, the proportion of classes in regions of the feature space changes. The only difference between conditions 1 and 3 is that the latter is free of the restriction of preserving $P(X)$.

Condition 2 states that the characteristics that define each class change. However, $P(Y)$ must be preserved, while $P(X)$ can change. $P(Y)$, on its own, dictates the proportion of the classes considering all data points. The only difference between conditions 3 and 4 is that the latter is free of the restriction of preserving $P(Y)$. For that reason, condition 4 is similar to the prior probability shift with the aggravating factor that the characteristics of each class have changed.

On a side note, concept drift type 1 is often called *virtual drift*. According to [Moreno-Torres et al. \(2012\)](#), concept shift is the hardest type of shift to deal with. Not only that, concept shifts 3 and 4 are rarer and possibly impossible to tackle. On the other hand, the two first shifts are easier. However, we find no reason to believe that concept shifts 3 and 4 are rare. A simple global linear transformation that moves all data points along some direction can cause concept shift 3 and, given a change in the proportion of classes, shift 4 as well. A linear transformation is illustrated in [Figure 18](#).

While such drifts may be impossible to deal with in typical batch learning problems, some assumptions can reverse such an unfavorable situation. For instance, changes in a stream can be incremental and therefore traceable over time ([DYER; CAPO; POLIKAR, 2014](#); [SOUZA et al., 2015a](#); [SOUZA et al., 2015b](#)), or can always lead to a previously known possible distribution for the data ([REIS et al., 2018b](#)).

Finally, we contest the easiness of concept shift one: in fact, this type of drift is impossible to detect in unsupervised settings, since we can only observe $P(X)$ and it does not change ([ŽLIOBAITĖ, 2010](#)). This fact is visually illustrated in [Figure 20](#).

Additionally, in some situations where drift may be undetectable when comparing observation windows that are distant from each other, the drift may be detectable by analysing intermediate windows. If there are intermediate drifts between these distributions and with small enough observation windows, it may be possible to trace the evolution of the drift over time and detect it without true labels ([DYER; CAPO; POLIKAR, 2014](#); [SOUZA et al., 2015a](#); [SOUZA et al., 2015b](#)). [Figure 21](#) illustrates such a case. Notice that this scenario is impossible if $\mathcal{X}$ is univariate: swapping the distributions of the classes turns the drift undetectable even with intermediate distributions, since it is impossible for two distributions to gradually swap their positions without ever overlapping in a one-dimensional space.

[Kull and Flach \(2014\)](#) extend the work of [Moreno-Torres et al. \(2012\)](#) by introducing graphical notations of the aforementioned types of dataset shifts, and 12 new

Figure 20 – This figure illustrates geometrically a concept drift that is undetectable without true labels in a two-dimensional feature space with two classes. The red-shaded area represents the boundaries of class c_1 , and the blue shaded-area the boundaries of class c_2 . Consider that there is **no class overlap**, that is, none of the data points that belong to c_1 are within the area of c_2 , and vice-versa.

Source: Elaborated by the author.

Figure 21 – Illustration of a case of traceable incremental drift. There are two classes, distinguishable by their unique color. This illustration presents five snapshots that represent the evolution of the data over time. The thick arrow represents a passage from one snapshot to the following. The curved arrow represents the movement of the class in the feature space. A brighter class with a dashed border represents the previous position. Notice that if we only compare t_0 with t_4 , we have a case that approximates $P_A(Y|X) \neq P_B(Y|X)$ and $P_A(X) = P_B(X)$, which is undetectable without true labels. However, with the support of the intermediate distributions, we can geometrically trace evolution of the data and therefore distinguish both distributions.

Source: Elaborated by the author.

additional sub-types of shifts for even further detail. As such granularity is unused in this research project, we point the interested reader to this paper for further information on this topic. Conversely, [Kelly, Hand and Adams \(1999\)](#) offers a simplified view that is often enough to specify a problem of changes in data. According to them, we name *concept drift* any changes in one of the following probabilities:

prior probability $P(Y)$ corresponds to the proportion of the classes;

conditional $P(X|Y)$ corresponds to the probability of an observation presenting certain behavior, given that it belongs to a certain determined class. A change in this probability means that the behavior associated with a class changed;

conditional *a posteriori* $P(Y|X)$: corresponds to the chance of a data point having a certain class, given the behavior it displays. A change in this probability is due to one or both of the previous probabilities, and is also called *real drift*. This probability is typically estimated (implicitly or explicitly) by classifiers.

Literature offers a great granularity for types of dataset shifts, and even methods that attempt to measure the intensity and direction of certain changes ([SARNELLE et al., 2015](#)). In practical terms, however, the methods discussed in this research project do not attempt to specify which type of concept shift has occurred. Instead, they are designed only to either perceive that $P(X)$ has change, or to precisely measure $P(Y)$. For this reason, we henceforth will refer to changes in data simply as changes in the distribution of the features or as changes in the distribution of the classes, referring to $P(X)$ and $P(Y)$, respectively.

Detecting that a change has occurred can carry different meanings and consequences depending on the application. For instance, in an application where there is interest in detecting the emergence of new classes of data (also called novelties), a change where a new cluster of data points appear may represent a novelty. [Figure 22](#) illustrates different practical types of change that occur in data stream literature. In this example, circles represent data points, and colors represent classes. The figure also shows the boundaries that separates the classes.

Specifically in applications where the task at hand is to identify the classes of data points, the boundaries of the classes may change over time. Hence, knowledge that is obtained with an initial set of data may become outdated and start hindering our performance at this task at a later time.

In the next section, we further discuss the tasks that are affected by changes in data, and that are object of study in this research project.

Figure 22 – Some types of changes in data that are frequently found in literature. Dashed lines indicate the separation margin between the classes.

Source: Souza *et al.* (2020).

3.2 Machine learning tasks

While, for some tasks, such as *classification*, changes in data appear as an aggravating factor that makes the problem more difficult to tackle, other tasks such as *quantification* only exist due those changes. Different tasks make different assumptions regarding data availability. In this work, the tasks studied particularly vary their assumptions regarding the available knowledge about class labels. We define and briefly discuss several tasks in this section, highlighting the role of data shift in their operation.

3.2.1 Classification

Classification is a machine learning task whose objective is to predict the class labels of *unlabeled* data points, which are those that have their true class label unknown to us.

To achieve this task, we are provided with a *labeled* training data L to induce a *classification model* h , as formalized in [Definition 4](#).

Definition 4. A **classifier** is a model h induced from L such that

$$h: \mathcal{X} \longrightarrow \mathcal{Y}$$

which aims to predict the classes of unlabeled observations correctly.

Given that the test sample U contains all unlabeled data points, the objective of the classifier is to produce $h(U_X) \simeq y(U_X)$, where $y(U_X)$ is previously unknown.

In practice, h learns how to model $P_L(X, Y)$, the probability distribution of the data that generated the observation points in the training set L . Therefore, if $P_L(X, Y) \neq$

$P_U(X, Y)$, that is, if the unlabeled data in U is sampled from a different probability distribution than the training data in L , the correlations between the feature space and the existing classes that were learnt by the classifier may be unrepresentative of the ones that exist in U . Consequently, h will have a considerably worse performance (PAN; YANG, 2009; QUIONERO-CANDELA *et al.*, 2009; SAENKO *et al.*, 2010; BEN-DAVID *et al.*, 2007).

3.2.2 Scoring

There are different mechanisms employed by classifiers to decide which class will be assigned to any given observation. Many of them take this decision by estimating and comparing values that correlate with the posterior probabilities $P(Y|X)$.

We emphasize one that is frequently adopted for binary classification problems, that is, problems where $|\mathcal{Y}| = 2$. In binary classification, one of the two classes is denominated *positive class* ($c_1 = c_+$). The other one is denominated *negative class* ($c_2 = c_-$). In this setting, one can induce a classification model capable of issuing a *score* $h^s(x)$ for a given observation x . If such a score is greater than a certain threshold t_h , the observation is classified as positive. Otherwise, it is classified as negative (FLACH, 2012). To shorten our text, we henceforth refer to scores of negative observations simply as negative scores, and analogously refer to scores of positive observations as positive scores. Such denominations are not to be confused with the sign of the numerical value of the scores. One can formalize classification with a scorer as follows:

$$h(x) = \begin{cases} c_+, & \text{if } h^s(x) > t_h \\ c_-, & \text{otherwise} \end{cases}$$

By separating the task of issuing a score from the classification task, we can formalize a *scorer* according to Definition 5.

Definition 5. A *scorer* is a model h^s induced from L such that

$$h^s : \mathcal{X} \longrightarrow \mathbb{R}$$

which produces a number that correlates with $P(c_+|x)$.

For multiclass problems, classifiers often output a *class-confidence* $h^v(x, c)$ for each combination of data point and class. The class that maximizes the class-confidence is assigned to its corresponding observation. The classifier's confidence, hereafter simply called *confidence*, is the maximum class-confidence for an observation point. The following equation formalizes such process.

$$h(x) = \arg \max_{c \in \mathcal{Y}} \{h^v(x, c)\}$$

We emphasize the distinction between *score* and *confidence*, since both terms will be used distinctively throughout this document. The *score* is a value that correlates with the estimated probability of an instance belonging to the positive class, in a binary classification problem, *i.e.*, the $h^s(x)$ correlates with $P(y(x) = c_+) = P(c_+|x)$. A *class-confidence* is a value that correlates with the estimated probability of an instance belonging to a given class, in a multiclass problem, *i.e.*, $h^v(x, c)$ correlates with $P(y(x) = c) = P(c|x)$. The *confidence* of the classifier is the maximum class-confidence, *i.e.*, the confidence for a given observation point x is $\max_c \{h^v(x, c)\}$, which correlates with $\max_c \{P(c|x)\}$.

Since scoring involves estimating values that correlates with the posterior probabilities $P(Y|X)$, a real drift evidently disrupts the modeling properties of scorers.

3.2.3 Quantification

In the Machine Learning community, a task of growing interest is Quantification (GONZÁLEZ *et al.*, 2017). The objective of this task is to estimate the proportion of each class, or class distribution, in an unlabeled test sample. Such a goal is found in a number of practical applications from a wide range of research fields and areas. For instance, in social sciences, quantification is used to predict election results by analyzing different data sources that support the candidates (HOPKINS; KING, 2010). In natural language processing, quantification estimates the prior probability of different senses for a given word (CHAN; NG, 2006). In entomology, it infers the local density of mosquitoes in a specific area covered by an insect sensor (CHEN *et al.*, 2014).

Although quantification and classification share similar characteristics, the main one being the representation of data, their objectives differ. A quantifier need not provide individual class predictions. Instead, it must assess the overall quantity of observations that belong to a specific class or a set of classes (GONZÁLEZ *et al.*, 2017). A quantifier is formally defined in Definition 6.

Definition 6. A **quantifier** is a model that predicts the prevalence of each class in a sample, such that

$$q: \mathcal{X}^? \longrightarrow [0, 1]^l$$

$\mathcal{X}^?$ denotes the universe of possible samples from $\mathcal{X}$, with an arbitrary number of observation points. For a given unlabeled test sample $U \in \mathcal{X}^?$, the quantifier outputs a vector $\hat{Q} = [\hat{P}(c_1), \dots, \hat{P}(c_l)]$, which estimates the prior probability for each class, such

that $\sum_{j=1}^l \hat{P}(c_j) = 1$. The objective is $[\hat{P}(c_1), \dots, \hat{P}(c_l)]$ to be as close as possible to the true prior ratios $[P(c_1), \dots, P(c_l)]$ of the probability distribution U was sampled from.

As the main task is to measure the prior probabilities of the classes in U , it is implied that the class distribution changes significantly from the labeled training sample L (which supports the induction of q) to the unlabeled test sample U .

One straightforward way of achieving quantification is to count the predictions produced by a classifier. This method is called Classify and Count (CC) (FORMAN, 2005). Naturally, performing CC with a perfect classifier always produces a perfect quantification. However, accurate quantifiers do not necessarily need to rely on accurate classifiers. Since our objective is purely to count how many observations belong to each class, misclassifications can nullify each other, as illustrated in Table 4.

Table 4 – Confusion matrix of a fictional classification model that achieved 77% accuracy given a test sample. Although the model was not a perfect classifier, it obtained perfect quantification in the presented scenario: it predicted the number of positive observations to be 1,000, which is the correct amount – false positives and false negatives cancelled out.

		Prediction		
		Positive	Negative	Total
Actual	Positive	830	170	1000
	Negative	170	330	500
Total		1000	500	1500

Source: Elaborated by the author.

Figure 23 illustrates a scenario where we obtain perfect quantification regardless of imperfect classification, given a classification model based on a scorer. This illustration will come in hand to visually understand the systemic error of CC, afterwards.

Nevertheless, CC is systemically flawed for any imperfect classifier. We accentuate the fact that perfect classifiers are rarely achievable for real-world applications. For illustration purposes, consider a case of binary quantification. Let $\hat{p} = \hat{P}(c_+)$ be the assessed proportion of the positive class for a test sample, while $p = P(c_+)$ is the true positive class ratio. $P_h(\oplus|\ominus) = P(h(\mathbf{x}) = c_+ | y(\mathbf{x}) = c_-)$ is the classifier's False Positive Rate (FPR), that is, the proportion of negative observations that are wrongly classified as positive. Analogously, $P_h(\oplus|\oplus) = P(h(\mathbf{x}) = c_+ | y(\mathbf{x}) = c_+)$ is the classifier's True Positive Rate (TPR), that is, the proportion of positive observations that are correctly classified as such. In this context, $\hat{p}$ is:

$$\hat{p} = pP_h(\oplus|\oplus) + (1 - p)P_h(\oplus|\ominus) \quad (3.1)$$

Similarly, we can write the true positive class ratio as follows:

Figure 23 – Illustration of a quantifier that relies on a scorer-based classifier. In this illustration, the probability density functions of the scores for both classes are depicted. The density functions are scaled to depict the proportion of the classes. For the given threshold, we have a nearly perfect quantification, albeit classification is not perfect. The classification errors nullify each other.

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

$$p = pP_h(\oplus|\oplus) + pP_h(\ominus|\oplus)$$

From the two previous equations, we can derive that the absolute quantification error $|\varepsilon|$ committed by CC is:

$$\begin{aligned} |\varepsilon| &= |\hat{p} - p| \\ &= |pP_h(\oplus|\oplus) + (1-p)P_h(\oplus|\ominus) - pP_h(\oplus|\oplus) - pP_h(\ominus|\oplus)| \\ &= |(1-p)P_h(\oplus|\ominus) - pP_h(\ominus|\oplus)| \\ &= |pP_h(\ominus|\oplus) - (1-p)P_h(\oplus|\ominus)| \end{aligned}$$

where $P_h(\ominus|\oplus) = P(h(\mathbf{x}) = c_- | y(\mathbf{x}) = c_+)$ is the classifier's False Negative Rate (FNR), that is, the proportion of positive observations that are wrongly classified as negative.

Observe that the error relates to the absolute difference between the hatched areas (false positive and false negative) in [Figure 23](#). This fact may lead to the intuition that, for a score-based quantifier, it is enough to select a threshold that causes the number of false-negative observations to be the same as the number of false-positive observations. However, those values depend on the true-positive ratio, which is the variable we want to predict in the first place. Observe that if we ignore the absolute, we can easily express ε as a linear function of p :

$$\begin{aligned} \varepsilon(p) &= pP_h(\ominus|\oplus) - (1-p)P_h(\oplus|\ominus) \\ &= (P_h(\ominus|\oplus) + P_h(\oplus|\ominus))p - P_h(\oplus|\ominus) \\ &= \alpha p + \beta \end{aligned} \tag{3.2}$$

This implies that $|\varepsilon(p)|$, the absolute quantification error committed by CC, grows linearly when the actual positive class ratio p is under or above a certain value for which quantification should be perfect, in any case where either $P_h(\oplus|\ominus)$ or $P_h(\ominus|\oplus)$ is not null. Figure 24 illustrates this effect with a real dataset.

Figure 24 – Experimental performance of a quantifier that relies on a scorer-based classifier for several true positive ratios. The continuous blue curve is the actual absolute quantification error (AQE) produced by the model, and the red dotted curve is the absolute prediction of AQE (Equation 3.2) given FPR and TPR estimated via 10-fold cross validation. The classifier is a Random Forest with 100 trees. Score is given by how many votes the positive class received. Dataset (Insects v2, described in Section 5.3.6) includes flight information for female *Aedes aegypti* (positive class) and female *Culex quinquefasciatus* (negative class).

Source: Research data.

Figure 25 further illustrates the effect on the density function of scores in a score-based classifier. In general, if the proportion of positive observations is greater than the one that causes perfect quantification, the predicted positive ratio is underestimated since the number of false negatives becomes greater than the number of false positives. Analogously, if the proportion of positive observations is lower than the one that causes perfect quantification, the predicted positive ratio is overestimated. We point the interested reader to (TASCHE, 2016) for a thorough investigation on the limitations of quantification without adjustments.

If one extends our analysis on binary CC to the multiclass scenario, a similar systemic error pattern would be found.

Although most quantification algorithms rely, at some point, on classifiers or scorers, there are several ways to minimize the systemic error. In the binary case, if we rewrite Equation (3.1) to isolate the true positive ratio p , we have:

$$p = \frac{\hat{p} - P(\oplus|\ominus)}{P(\oplus|\oplus) - P(\oplus|\ominus)} \quad (3.3)$$

The previous equation suggests that, if we can reliably estimate TPR and FPR, we can make an adjustment $\hat{p}$ that fixes the bias. This is the principle of Adjusted Classify

Figure 25 – Illustration of a quantifier that relies on a scorer-based classifier. This illustration shows the probability density functions of the scores for both classes. The density functions are scaled to depict the proportion of the classes. Contrary to Figure 23, the quantification is not perfect with the current proportion of positive observations, even though the classification threshold and the probability distributions for each class, taken individually, are all the same.

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

and Count (ACC) (FORMAN, 2005), that is defined in the following equation, where $\widehat{\text{TPR}}$ and $\widehat{\text{FPR}}$ are estimates for TPR and FPR, respectively:

$$\text{ACC}(\hat{p}, \widehat{\text{TPR}}, \widehat{\text{FPR}}) = \min \left\{ 1, \frac{\hat{p} - \widehat{\text{FPR}}}{\widehat{\text{TPR}} - \widehat{\text{FPR}}} \right\}$$

As ACC comes from Equation 3.3, it produces perfect quantification when the estimates of FPR and TPR are both correct. $\widehat{\text{FPR}}$ and $\widehat{\text{TPR}}$ are typically estimated with labeled training data and procedures as k -fold cross-validation.

Another way of achieving the same adjustment, and that is easier to extend to multiclass problems, is to write a linear system that makes explicit how each $\hat{P}(c_i)$ is estimated by CC. In the case of binary quantification:

$$\begin{bmatrix} P(c_+|c_+) & P(c_+|c_-) \\ P(c_-|c_+) & P(c_-|c_-) \\ 1 & 1 \end{bmatrix} \begin{bmatrix} P(c_+) \\ P(c_-) \end{bmatrix} = \begin{bmatrix} \hat{P}(c_+) \\ \hat{P}(c_-) \\ 1 \end{bmatrix}$$

3.2.4 Open-set Quantification

Several real-world applications in quantification and classification involve an overwhelming large number of classes, such as diagnosing patients with diabetes (CLAESEN *et al.*, 2015), detecting tweets that promote hate and extremism (SUREKA; AGARWAL, 2014) and counting the number of insects that belong to a target species in a region. Understandably, gathering data that discriminate all existing classes in such examples is a virtually impossible task. Data should be collected for all other existent diseases, all

other possible sentiments a tweet can carry, and all other insect species. Such goals are clearly unattainable.

In practical terms, recognition models for these application domains operate in an *open-set* scenario, in which only part of the classes are known during training, but additional classes may be observed after the model is put into operation. This setting contrasts with the typical *closed-set* scenario, in which all classes that are observed after the model's deployment are expected to have been previously incorporated into training data.

Let us consider the case in which we want to quantify the observations of one class of interest. Data for other negative sub-classes available constitute our partial knowledge about the negative class. We emphasize that although the negative training data can comprise several sub-classes, we can face situations where no single attribute differentiate between them.

A straightforward approach to tackle open-set quantification is to adopt off-the-shelf multi-class or binary quantification algorithms. Nevertheless, in the open-set scenario, performance statistics that standard quantification algorithms rely on, as FPR, cannot be reliably estimated from training data. In fact, all of such quantification algorithms rely on data derived from the distribution of the negative class estimated with a training set. Yet, in the presented case, the training set is nonetheless unlikely to be representative of reality.

The negative class is a mixture of different distributions, due to its sub-classes. By the very definition of the quantification problem, we are unable to predict the degree of contribution of each distribution in the test data using training data. [Figure 26](#) illustrates the variety of sub-class distributions with real-world data.

In [Figure 26](#), scores for female *Culex quinquefasciatus* overlap with scores for female *Aedes aegypti* more than the other classes do. This means that it is harder to differentiate between female *Aedes aegypti* and female *Culex quinquefasciatus*. Therefore, if the presence of *Culex quinquefasciatus* in the test set increases, then the same happens to the number of false positive, false negative or both (depending on where a threshold is set). However, the proportion of female *Culex quinquefasciatus* is previously unknown in test data so that we cannot set expectations for false positives or false negatives ahead of the test sample.

Multi-class quantifiers, as the multi-class ACC, can deal with this uncertainty regarding FPR and TPR (which are binary classification statistics), *if* we can accurately estimate misclassification rates between *all* pairs of classes. Indeed, in the previous example, despite not knowing the proportion of each negative sub-class in the test data, we still have individual training data for each and all of them. However, two issues persist. First, multiple sub-classes can potentially be erroneously grouped together. In this

Figure 26 – Score distributions produced by a binary classifier trained with a dataset where the positive class contains female *Aedes aegypti* mosquitoes and a negative class is composed of male *Aedes aegypti*, and male and female *Culex quinquefasciatus*. On top are the score distributions for a test set where the negative class contains only male *Aedes aegypti*. Middle are the score distributions for a test set where the negative class contains only male *Culex quinquefasciatus*. Bottom are the scores for a test set where the negative class contains only female *Culex quinquefasciatus*. The classifier is a Random Forests with 200 trees.

Source: Research data.

example, female and male insects from a same species could have been analyzed without distinction, even though they behave differently. Second, specifically in an open-set scenario, we expect training data to lack observations from sub-classes that will nonetheless be present after the model is deployed. Figure 27 illustrates a situation where not all negative sub-classes are present in training data, and the corresponding effect.

Suppose that, in Figure 27, we are only interested in counting the positive observations. Although we are not interested in quantifying the unseen class (female *Culex quinquefasciatus*), it can significantly hinder the performance of the quantifier if it is present in the test data. In the worst case, if the test sample comprises only this particular species, the quantifier would inform that the observations are mostly female *Aedes aegypti*. We note that, in a typical open-set application, we may rely on the expectation that the training set will contain some class that is similar to female *Culex quinquefasciatus*. However, such an expectation might be unrealistic.

3.2.5 Positive and Unlabeled Prior Estimation

Recall that open-set methods that make use of available data for any negative sub-class have to assume at least some sort of correlation between training and test samples. Consequently, if the universe contains an exceedingly large number of sub-classes, the ones for which we have data might not be enough to reliably model the general behavior of the negative class. In such cases, the negative class remains, for practical purposes, unpredictable.

Figure 27 – Score distributions produced by a binary classifier trained with a dataset where the positive class contains female *Aedes aegypti* mosquitoes and a negative class is composed of male *Aedes aegypti*, and male *Culex quinquefasciatus*. On top are the score distributions for a test set where the negative class contains only male *Aedes aegypti*. Middle are the score distributions for a test set where the negative class contains only male *Culex quinquefasciatus*. Bottom are the scores for a test set where the negative class contains only female *Culex quinquefasciatus*, which is *not* included in the training set. The classifier is a Random Forests with 200 trees.

Source: Research data.

In addition to that, there are legitimate cases where it is impractical to collect samples with only negative observations. For example, suppose that we sell a product online and we can track the preferences of our customers via their social media profiles. Our customers can be used as positive training data for the task of identifying who might be interested in purchasing our product. On the other hand, gathering data for the negative class is not as trivial. If we randomly sample online social media profiles, the resulting set would contain both people who are uninterested in the product and potential customers. An explicit data gathering for the negative class could involve an online poll, which is time consuming and can still generate potentially biased data.

In a second example, suppose that we want to count the number of people that are infected with a disease in a population. Due to procedure costs, people may test for a disease only if they are suspicious of having it. In that case, while we can have a sample of people that were positively tested for such a disease, our data for people who were negatively tested may be severely lacking and biased. In such a case, a random sample of people would include both people who are not infected and people who are infected but were never diagnosed.

If we are interested in quantifying one single positive class and we are unable to have a reliable representation of the negative class alone, we may need to rely solely on positive training data to induce a quantification model. Any available data regarding the negative class can instead be used to evaluate such a model, although we emphasize that the evaluation is still limited since such data misrepresent the negative class.

Positive and Unlabeled Prior Estimation (PU PE) is a task derived from Positive and Unlabeled Learning (PU L). The main task of the latter is akin to classification. To better explain PU PE, we first briefly introduce PU L.

In the general case of PU L (ELKAN; NOTO, 2008), we are provided with two samples of data. One of such samples, L , contains only positive (and therefore labeled) observations, whereas the other, U , contains unlabeled observations that can be either positive or negative. The objective is to infer the individual labels of the observations in the unlabeled sample. Figure 28 illustrates the general setting of PU L.

Figure 28 – Illustration of the general setting of Positive and Unlabeled Learning in a two-dimensional feature space. The filled (blue) circles correspond to the exclusively positive data sample, and the unfilled shapes correspond to the unlabeled data sample. In the unlabeled data sample, circles are positive observations, and squares are negative observations. However, such labels in the unlabeled sample are not provided to the PU task at hand.

Source: Elaborated by the author.

Observe that the basic description of PU L does not pose explicit restrictions regarding the proportion of the classes in the unlabeled data. However, possessing the information of such a statistic make the labelling task an easier problem (ELKAN; NOTO, 2008). If the labelling is based on a scorer, for instance, the number of positive observations can be used to set a threshold for it. Therefore, PU Prior Estimation is a sub-task and has the sole objective of predicting the proportion of the classes, which could eventually support later labelling.

A common assumption across different pieces of work on the area is that the labeled sample is “selected completely at random”. More specifically, it states that each positive observation has a constant probability c of being labeled (ELKAN; NOTO, 2008;

BEKKER; DAVIS, 2018). Consider s a function that annotates whether a positive observation is labeled, as follows:

$$s(\mathbf{x}) = \begin{cases} 1 & \text{if } y(\mathbf{x}) = c_+ \text{ and } \mathbf{x} \text{ is labeled} \\ 0 & \text{otherwise} \end{cases}$$

In such a case, the assumption specifies that

$$c = P(s(\mathbf{x}) = 1 | y(\mathbf{x}) = c_+) = P(s(\mathbf{x}) = 1 | y(\mathbf{x}) = c_+) \quad (3.4)$$

from which follows (ELKAN; NOTO, 2008)

$$P(s(\mathbf{x}) = 1) = cP(y(\mathbf{x}) = c_+) \quad (3.5)$$

In a simplification, the labeled sample is one that *could plausibly have been* a uniform sample from all available positive observations. More importantly, this assumption and how it is exploited by the algorithms underline that the labeled sample and the positive observations from the unlabeled sample share the same probability distribution. Therefore,

$$P(\mathbf{x} | s(\mathbf{x}) = 1) = P(\mathbf{x} | s(\mathbf{x}) = 0, y(\mathbf{x}) = c_+) = P(\mathbf{x} | y(\mathbf{x}) = c_+).$$

3.3 Final considerations

In this chapter, we reviewed the most relevant concepts that are used throughout this work. In summary, we are interested in analyzing samples that contain events that are (mostly) independent of each other, albeit sharing a same background distribution. The samples can be slices of an ongoing streaming data (in the form of sliding windows) or individual batches of data without any particular order. The probability distribution of the data points is prone to change over time due to dataset shift. Finally, each data point has an associated class label, which is unknown for the test data.

We are mainly interested in two related, although distinct tasks: classification and quantification.

In the general case of classification, we aim to predict the true label for each individual data point. In this particular work, we assume that, although the distributions of different test samples may vary, they will always fall within a limited set of already known distributions, called *contexts*.

On the other hand, for quantification, we aim to predict the proportions of the classes within a sample, disregarding individual class-labels. In this work, we tackle quantification in two fronts: binary quantification, purely as a support activity for context identification, and one-class quantification, where training data include only observation points that belong to the positive class.

In the next two chapters, we thoroughly study each one of the aforementioned tasks, reviewing literature and proposing our methods.

CONTEXT IDENTIFICATION

The contents of this chapter are based on our contributions to published papers (REIS; MALETZKE; BATISTA, 2018; REIS *et al.*, 2018b; MALETZKE *et al.*, 2018; MALETZKE *et al.*, 2019) and additional material that is yet to be published.

In this chapter, we describe how we tackle the dataset shift problem through context identification. We propose a scenario in which we assume all possible test distributions can be analyzed prior to the observation of test samples. In this new setting, we introduce the task of context identification, which aims to identify the distribution of unlabeled test samples among those that were previously analyzed. After a distribution is identified, we can employ appropriate classification models.

Before detailing our methods, we discuss the related work and their similarities and differences to the scenario we are proposing.

4.1 Related work

The community has paying growing attention to the dataset shift problem and how it can affect real-world applications. Notwithstanding the necessity covering a wide range of real cases, the first attempts on the problem assumed instant availability of true class-labels right after the classification of each observation point in a stream (GAMA *et al.*, 2004; BIFET; HOLMES; PFAHRINGER, 2010; OZA, 2005; BIFET; GAVALDA, 2007a; BIFET; GAVALDÀ, 2009). As such an assumption hardly holds in practical problems, newly proposed methods target more realistic scenarios. We are, in this work, solely interested in the scenario where no true labels are ever obtained for test data points.

In this section, we investigate the alternative assumptions that were incorporated to mitigate the necessity of true labels.

4.1.1 Label latency

Verification latency in data streams, or delay, is the period between the observation of an unlabeled data point and the availability of its true label (MARRS; HICKEY; BLACK, 2010). This period depends on the application domain. For instance, if we are interested in predicting the tendency of the stock market or electricity demand, the verification latency is the prediction window, that is, how long ahead of time the prediction targets. In other applications, verification latency can vary over time. This is in the case of sensors in an external environment, as they require someone go to the site of the sensors, gather the data and an area expert to classify their events. This process may take different lengths of time depending on logistic concerns about when to gather the data and the amount of data collected.

Null-verification latency occurs when the true labels are available right away after the classifier makes a prediction. It is a rare phenomenon in real-world applications, since either **(a)** there must be a period of time between the prediction and the actual occurrence of the event, if we are predicting future observations ahead of time, or **(b)** there must be a period for the data point to be correctly classified by an external entity, such as a specialist on the field. In either way, the time between a prediction and the availability of its corresponding true label is that in which one can plan an appropriate reaction and execute it. For instance, if a classification system foretells a high electricity demand in a 30 minutes window, there are 30 minutes for planning and executing actions that will lead to an increase of electricity generation to attend the predicted demand.

Different publications tackle label availability with different assumptions and different experimental setups. In extreme label latency, true labels are available within a short slice at the beginning of the stream. Such labels are used to identify the classes and build an initial classification model. After that, no true labels are further recovered, and the approach needs to recognize concept drifts and adapt to them unsupervisedly (SOUZA *et al.*, 2015; DYER; CAPO; POLIKAR, 2014).

In contrast, semi-supervised learning and active learning settings assume label availability for just a portion of the data. Semi-supervised learning expects a usually small fraction of labeled data points while the majority is unlabelled. Although the labeled data could be observed with any latency, most of the literature assumes they are observed with null latency. Therefore, a typical setting for semi-supervised learning is to have null-latency for the labeled data and extreme verification latency for the unlabeled data.

Active learning also conjectures that only a portion of the data is labeled. The critical difference between active and semi-supervised learning is that, in the first approach, the algorithm can choose which events will be tagged by an oracle. Again, it is a common assumption in the literature that the oracle immediately labels all assigned examples.

There is a small body of work on streaming classification assuming extreme verification latency. For instance, concept drifts have been modeled as data clusters movements in the feature space (DYER; CAPO; POLIKAR, 2014; SOUZA *et al.*, 2015; FARIA *et al.*, 2016b), which limits these methods to deal with incremental drifts and nearly constant class distributions. To illustrate that, Dyer’s proposal (DYER; CAPO; POLIKAR, 2014) applies a technique of semi-supervised classification on consecutive batches of unlabeled data. For each batch, the predicted labels of the previous batch are assumed to be correct. Souza *et al.* (2015) periodically performs data clustering and checks the spatial similarity between the clusters to assume a movement for the data in the feature space. Similarly, Faria *et al.* (2016b) propose MINAS, which also use microclusters to incrementally update the model and identify novelties.

However, the co-occurrence of incremental drifts and extreme class distribution variation can mislead these methods. For instance, the insect sensor only registers the insects that pass in front of it. Hence, the chance of registering flights for a species depends on its presence and activity. The insect circadian rhythm governs its periods of activity, making it vary throughout the day. The chance of registering events for a species diminishes when it is inactive. When that species becomes active again, it can have a different data distribution than it had during the last active period. Therefore, it may be unfeasible to infer an incremental drift due to the lack of data between consecutive active periods. In addition to that, classification models may be trained with data previously observed in laboratory and only later deployed on the field, where actual streaming data is observed. In this case, an abrupt drift is assured.

Literature also tackles label scarcity and delays with semi-supervised techniques. Pozzolo *et al.* (2015) introduce a method that directly addresses delays in the availability of actual labels for credit card fraud detection. The proposal expects delayed labels for a significant portion of the data and readily available labels for a smaller part. The obtained results suggest that an ensemble including a classification model induced solely on the small labeled portion provides better results than a single model induced on the whole data.

Masud *et al.* (2010) also adhere to semi-supervised approaches. First, they introduce an algorithm that is capable of both tackling delays in the availability of real labels and detecting novelty, that is, the emergence of new classes. However, besides specific assumptions regarding the feature space, there is also the assumption of the possibility of delaying the prediction of observed events. In a follow-up work (MASUD *et al.*, 2012), they introduce a method for the true label scarcity problem, where actual labels are available for only a portion of the data. The labeled data are used in a semi-supervised fashion to classify future unlabeled events.

Wu, Li and Hu (2012) introduce another semi-supervised approach that tackles

label scarcity in data streams. They use a decision tree that incrementally grows while keeping groups in its leaves. Such groups detect concept drifts through deviations in stored statistics.

Kuncheva and Sánchez (2008), Amir and Toshniwal (2010) propose methods that directly tackle delays in the availability of true labels. However, they are limited to stationary distribution, *i.e.*, classification problems without concept drifts.

Finally, Gomes *et al.* (2017b) introduce a new data stream classifier. During the experimental evaluation, such classifier is tested against a specific setting of delay (one example's true label is made available after 1,000 other examples are classified).

Although the work listed in this section have an assumption that is generally more realist than null-latency, they are still restrictive and not applicable for several real-world applications, such as the sensor for insects. For one, solutions that track incremental changes in a stream cannot handle models that are trained once, in a laboratory, to be later employed in several and divergent situations. That use incurs an initial abrupt drift. Additionally, true label availability can be a plain impossibility outside the laboratory environment, making any semi-supervised approaches unsuitable.

4.1.2 *Unsupervised drift detection*

When no true labels are ever available, one possible approach is to detect statistical changes in the distribution of the data – $P(X)$. We, once again, emphasize that not all changes are detectable without true labels (ŽLIobaITĚ, 2010).

To the best of our knowledge, Kifer, Ben-David and Gehrke (2004) are the first to propose the direct application of hypothesis tests to detect concept drifts without true labels. In this case, a fixed sample of events represents a training set and a dynamic sample represents data that was recently observed. A hypothesis test is then applied to verify whether the distributions of these samples differ. They evaluated the adoption of five statistical tests, including Kolmogorov-Smirnov and Wilcoxon tests. The authors described an algorithm for the fast re-computation of a variant of the Kolmogorov-Smirnov statistics, that is, the proposed algorithm does not compute the exact Kolmogorov-Smirnov statistic according to its definition. The authors focused entirely on the task of drift detection and, as such, did not evaluate the impact on other downstream tasks, such as classification.

Using a similar approach, ŽliobaITĚ (2010) proposes a method that can identify detectable changes in a stream: a hypothesis test is performed to verify if two consecutive sliding windows follow different distributions. An attractive proposal of this paper is the use of the test over the classifier class-confidences, instead of the feature values. As most statistical tests are univariate, this idea simplifies the test application and the decision process of indicating when a drift occurred. Two non-intersecting consecutive sliding win-

dows follow the data stream. The windows can contain either information from the feature space or the class-confidence from a classifier. After the observation of each data point, the windows are updated – removing the information relative to the oldest observation and inserting the information of the newest one. The method reports a concept drift if the distributions of both windows differ, according to a hypothesis test.

When using information from the feature space, Žliobaitė suggests using the whole vector that describes a data point as a multivariate variable and, therefore, adopting an appropriate hypothesis test, or mapping it to a single value. In the case of using the classifier’s class-confidence, if the classifier cannot provide this information, the predicted label itself is used as a replacement.

She evaluated three hypothesis tests for the concept drift detection task, including the Kolmogorov-Smirnov test. Žliobaitė advocates its use since it is a non-parametric test. Non-parametric tests better suit classification scenarios, where the underlying distribution is a mix of different distributions. Particularly for data stream applications, concept drifts can potentially intensify the occurrence of a mixture of distributions. The author does not address possible actions to be taken as response to the detection of a drift.

Reis *et al.* (2016) introduce an exact and incremental version of the Kolmogorov-Smirnov test (IKS) that suits data stream applications. They evaluate the impact of concept drift detection in classification regarding accuracy and number of labels demanded to update the model. Their setting shares similarities with active learning since their method requires labeled data to retrain the models after a concept drift.

Additionally, Reis *et al.* (2016) proposed different strategies to further decrease the number of labels requested after the drift is detected, although they could not completely remove the need of true labels.

Korycki and Krawczyk (2019) improve on the work of Reis *et al.* (2016) by proposing the use of IKS with intelligently selected sub-spaces of the feature space that better describe the classification problem, effectively avoiding false drift detection.

Despite the satisfactory results at detecting drifts in literature, purely detecting changes, even if unsupervisedly, does not solve the problem of how to handle drifts after their detection, ceding the very first assumption of extreme verification latency. Additionally, all previous papers do not differentiate changes in $P(Y)$ from changes in $P(X)$, which may lead to the detection of drifts that do not necessarily lead to decrease in classification performance.

Another area of research known as Domain adaptation has proposed methods that can be used in a complementary way to unsupervised drift detection. Related work on this field is discussed in the next section.

4.1.3 Domain adaptation

In the terminology of Domain Adaptation, we are given labeled samples from source domains and unlabeled samples from target domains. Each domain represents a data distribution. The objective is to build models, using the data from the source domains, that are capable of correctly classifying data from the target domains. The body of work splits into two streams: one studies Single-source Domain Adaptation (SUDA), where there is only one source domain; the other stream studies Multi-source Domain Adaptation (MUDA), where there are multiple source domains (ZHU; ZHUANG; WANG, 2019).

Within SUDA, Al-Otaibi *et al.* (2015) introduce the Versatile Trees. Such trees can adapt to batches of data that are monotonic transformations of the original data used for training. One benefit of the method is that the tree can be induced only once. However, in addition to the strong assumption of monotonic transformation, the technique requires the proportion of classes of the test samples to be known. However, unpredictability of the proportion of classes is an assumption that we have in this work, making the use of Versatile Trees impractical.

Other common approaches for SUDA include reweighing data points (HUANG *et al.*, 2007; KUMAGAI; IWATA, 2017), reducing the feature space into an invariant subspace where source and target domains are closer (LONG *et al.*, 2015; SUN; SAENKO, 2015; BAKTASHMOTLAGH; HARANDI; SALZMANN, 2016), and finding unilateral transformations that project the source domain to the target domain (FERNANDO *et al.*, 2013; SUN; SAENKO, 2015; KUMAGAI; IWATA, 2019).

In relation to MUDA, one possible approach is to combine multiple sources and apply a SUDA method. However, doing so can incur temporal overlap, as explained in Section 3.1.4.

Xu *et al.* (2018), Zhao *et al.* (2018) develop methods that involve having one classifier for each domain and later voting the class-labels of data points from the target domain. Although this work is similar to ours in that sense, their proposals focus on extracting common domain representations of the feature space that are invariant for all domains, which may be unfeasible.

Zhu, Zhuang and Wang (2019) is similar to the work of (XU *et al.*, 2018; ZHAO *et al.*, 2018). However, they allow for domain-specific feature extraction to cover cases where invariant common domains are unfeasible. They limit their evaluation to only two benchmark datasets.

One interesting tool that is commonly used in Domain adaptation is the Maximum Mean Discrepancy (MMD) (KUMAGAI; IWATA, 2019; ZHU; ZHUANG; WANG, 2019). MMD compares the mean of two distributions after embedding each one into the

Reproducing Kernel Hilbert Space (RKHS). Effectively, MMD disregards the density of the distribution, which makes it insensitive to changes in $P(Y)$.

The MUDA methods listed here share some severe limitations. They rely on deep learning approaches to extract features and, in the case of (ZHU; ZHUANG; WANG, 2019), to align the final predictions (fundamentally weight the individual classifiers). These techniques are data hungry and need a considerable amount of time to learn. Additionally, the feature space must be large enough so that extraction and reduction may be feasible. These observations reflect the reduced number of benchmark datasets adopted in the papers. Another limitation is the high computational cost for *each* target domain: performing the costly learning procedure is needed for each one of them. If we are dealing with concept drift in a stream, from instance, a rapid response to multiple and different target domains is expected.

To refrain from having such limitations, in this work we add an assumption that allows us to use a more restrict number of data points and process several samples faster: at least one of the source domains has the same *feature* distribution as the target domain. However, the distribution of the classes can vary.

4.1.4 Binary quantification

Binary quantification, as a machine learning task, is not the focus of our contribution in this chapter. However, our proposal makes use of a quantification algorithm so that its explanation is appropriate.

HDy is a mixture model quantification algorithm (GONZÁLEZ-CASTRO; ALAIZ-RODRÍGUEZ; ALEGRE, 2013). In HDy, the distribution of each class is estimated as a normalized histogram (H_+ and H_- for the positive and negative classes, respectively). The normalization is so that the sum of the bins is one. As a consequence, mixtures of both distributions can be obtained as a weighted vectorial average of the histograms, where the sum of the weights is one and both weights are positive. The weight that multiplies H_+ (α) represents the proportion of the positive class, while the weight that multiplies H_- ($1 - \alpha$) represents the proportion of the negative class.

Given an unlabeled test sample that produces histogram H_U , the objective of HDy is to find the combination of weights that minimizes the Hellinger Distance (HD) between the mixture distribution and the unlabeled distribution. Figure 29 illustrates the application of HD inside HDy.

To estimate the histograms, HDy first induces a scorer from a training sample containing both classes. Later, such a scorer is applied on two validation sets, where one has only positive data points, and the other has only negative data. The scores from each validation set are summarized as histograms by splitting the range $[0, 1]$ into intervals

Figure 29 – 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: Elaborated by the author.

(bins) and counting the proportion of scores that fall into each one of them. Laplace's correction can be optionally used to avoid empty bins. When presented with an unlabeled test sample, the algorithm builds the test histogram by applying the same scorer on this sample. Formally, HDy is defined as follows:

$$\hat{p} = \text{HDy}(H_+, H_-, H_U) = \arg \min_{0 \leq \alpha \leq 1} \{ \text{HD}(\alpha H_+ + (1 - \alpha) H_-, H_U) \}$$

Hellinger Distance is a function that quantifies the dissimilarity between two probability distributions. Hellinger Distance is defined as follows, where b is the number of bins.

$$\text{HD}(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_i^b (\sqrt{p_i} - \sqrt{q_i})^2}$$

Although **HD** is a metric, the interpolation between H_+ and H_- used inside **HDy** is inadequate in the Hellinger space. In [Section 4.5](#), we empirically confirm that $\text{HD}(A + xB, C)$ is not unimodal. The authors of **HDy** suggest finding **HDy** with a linear search through α , after discretizing the interval $[0, 1]$. On the other hand, we suggest using Ternary Search even if the results could be sub-optimal, to speed-up the algorithm. We back the viability of this suggestion with an experiment in [Section 4.5](#).

There are two clear advantages in representing the distribution of the training data as a linear interpolation between two histograms. As scalable vectors in $\mathbb{R}^b$ represent the distributions, when we need to represent a more significant number of events for one of the classes, we do not need to duplicate individual events. In this way, all events still have the same weight. Similarly, when we need to represent a smaller number of events for the other class, we do not need to sample individual observations, and therefore, we do not discard useful information. Notably, when scaling down a class, if the number of available observations in the training set is low, discarding any information can be critical and further skew from the actual distribution of the data. However, depending on how

big b is in relation to the number of data points, and on which distance function is used to compare the histograms, we may face the curse of dimensionality.

Maletzke *et al.* (2019) generalizes HDy by replacing the Hellinger Distance with several alternatives (CHA, 2007; CHA, 2008; CHA; SRIHARI, 2002). We further discuss our contributions in that regard in Section 4.6.

4.2 Practical motivation

In this section, we present initial experiments that provide reasoning to back the approaches used in the remaining of this chapter.

WBF-Insects is a simplistic yet real dataset that was collected in laboratory. This petite version of the sensor data comprises only two features for each data point: the wing-beat frequency (WBF) of the insect passing in front of the sensor and the ambient temperature at the instant of time when the passage was registered. There are records for three different species: *Aedes aegypti*, *Aedes albopictus* and *Culex quinquefasciatus*. *Aedes aegypti* and *Aedes albopictus* are additionally discriminated by sex, totalling five classes.

Temperature influences the metabolism of insects and, consequently, their wing-beat frequency (TAYLOR, 1963; MELLANBY, 1936). In this experiment, we consider the temperature as a possible latent variable for a classification task, where we only have access to it in a period of time previous to the actual use of the predictive models. This assumption is following a practical application of sensors: it is possible to collect more information in the laboratory than in the field, where we can be constrained by financial and scalability limitations.

We divided the dataset into two contexts, C_A , and C_B . C_A contains data points observed under below 25° Celsius, while C_B contains observations made under temperatures greater or equal than 25° Celsius. After the split, we removed the temperature feature from the data. In Table 5, we quantify the number of data points for each class and context.

Table 5 – Number of events for each class and context in the **WBF-Insects** dataset.

Class	C_A	C_B	$C_A \cup C_B$
1	2068	4942	7010
2	4297	6001	10298
3	6921	15511	22432
4	562	1379	1941
5	36852	22867	59719
Total	50700	50700	101400

Source: Reis, Maletzke and Batista (2018).

We show, in [Figure 30](#), that both contexts present distinct, although similar, probability distributions. Hence, we expect that a non-parametric hypothesis test such as the Kolmogorov-Smirnov test is capable of distinguishing them, given enough observations. Additionally, and more importantly, we can use the statistic of a given hypothesis test as a dissimilarity measurement to identify which of these two distributions is more similar to a third one.

Figure 30 – Estimated densities for the probability distributions of contexts C_A and C_B .

Source: Adapted from [Reis, Maletzke and Batista \(2018\)](#).

We randomly removed 300 observations from each context as training sets: L_A and L_B , from the contexts C_A and C_B , respectively. The remaining data points were used to synthesize a data stream. The observations were ordered so that the resulting stream has interleaved sequences of data points from C_A and from C_B . Within each sequence of observations from a same context, the data is shuffled. To some extent, this distribution of data is comparable to the real world: temperature change incrementally over time, so that it makes sense to group observations within a same context together.

We computed the Kolmogorov-Smirnov statistic between a sliding window containing 300 points that iterated over the stream, and both training sets (L_A and L_B). We present our results in [Figure 31](#), in which dashed vertical lines isolate the contexts, light blue background indicates context C_A , light red background indicates context C_B , the blue series represents the Kolmogorov-Smirnov statistic from comparing the sliding window with L_A , and the red series represents the Kolmogorov-Smirnov statistic from comparing the sliding window with L_B .

Generally, the red series has lower values than the blue series when the current context is C_B , and the blue series has lower values than the red series when the current context is C_A . Hence, we can identify which is the current context without temperature information, even if temperature is the variable responsible for splitting the data into such two contexts in the first place.

Figure 31 – Kolmogorov-Smirnov statistic over time for the WBF-Insects data stream.

Source: Research data.

Identifying which is the current context would be devoid of usefulness if the overlap among the classes in the feature space, for at least one of the contexts, is no smaller than the class overlap for the whole data without context separation. To measure the overlap of the classes, we used the measure called Intersection (CHA, 2008), since we are dealing with only one feature. It is a useful lower bound for the error, as it estimates the minimum misclassification probability. For a pair of normalized histograms with b bins, the Intersection is computed as follows:

$$\text{Intersection}(A, B) = \sum_{i=1}^b \min \{A_i, B_i\}$$

In other words, if a classification model h_{AB} induced over $C_A \cup C_B$ performs at least as well as individual classifiers for C_A and C_B , there is no reason to identify which is the actual context, since it would be enough to always issue predictions with h_{AB} .

To compute the Intersection, we adopted histograms of 10 bins to estimate the distribution of WBF for each class. When contexts were mixed together, *i.e.*, $C_A \cup C_B$, we obtained an intersecting area of 31%, which represents an upper bound for optimal classification accuracy of 69%. Histograms for the individual contexts C_A and C_B presented

intersecting areas of 15% and 21%, respectively, which represent an upper bound for optimal classification accuracy of 82%. These values indicate that it is beneficial to use two classification models and identify which is one that should be used.

We note that the *WBF-Insects* data is overly simple, since it has only one available feature, besides the context attribute. Introducing more features incur two problems. First, simply analyzing the data overlap and inducing expectations for accuracy becomes more difficult. Second, most hypothesis tests are unidimensional; therefore, we need a strategy to test for multiple features. There are different ways of circumventing this problem. We discuss two alternatives:

- **Applying a multidimensional hypothesis test:** although this may seem the most natural option, it carries many caveats. The most important one is the dimensionality of the feature space. Large feature spaces require bigger samples to achieve reasonable quality for the tests; Also, most of known tests are univariate. Another glaring caveat is the intrinsic assumption that all features are relevant for the underlying task (classification, in this case);
- **Applying a filter to reduce the dimensionality of the feature space to one:** one problem with this option is choosing the filter. The weights for different features should be accounted for so that they best reflect the classification models. However, classification models for different contexts may weight the features differently.

The first option is too compromising in practice: we would prefer to use the smallest viable number of recent observations to be more reactive to recent changes in data distribution. Thus we can often not afford to gather a sample that is large enough so that the multidimensional test becomes reliable.

One way of achieving the second option is to use the output from the classification models. Most classifiers output class-confidence, and the ones that do not can be ensembled or boosted in order to do so. In this case, training data can be gathered with validation sets or validation techniques such as leave-one-out and k -fold cross validation.

Another limitation of this small experiment is that the data has **constant** class distribution for each context during the whole stream. This situation is unrealistic, and we reinforce that the opposite is expected in our motivational application.

We observe that Kolmogorov-Smirnov statistic – as any other metric to *strictly* compare probability distributions – is sensitive to changes in the prior probability of the classes. To be precise, a probability distribution is defined not only by which values are observed, but also by their frequency. Therefore, variations in the proportion of the classes can severely affect the statistic measured and turn it unable to correctly differentiate the context.

In the next section, we propose methods to overcome such limitations.

4.3 Our proposals

In this section, we propose two methods to tackle the context identification problem. A context identifier is defined as follows:

Definition 7. A **context identifier** is a model that predicts which context, among those in a set, generated a given sample of data points, such that

$$c : \mathcal{X}^? \longrightarrow \mathcal{C}$$

$\mathcal{X}^?$ denotes the universe of possible samples from $\mathcal{X}$, with an arbitrary number of observation points, and $\mathcal{C} = \{C_1, \dots, C_z\}$ is the set of possible contexts.

We observe that context is not well defined. Indeed, the exact definition of context can vary from application to application. A plausible definition would equate context with probability distribution. However, as noted in the previous section, a probability distribution is a strict description of the behavior of data, which includes, besides the general shape that constrain data points in the feature space, their local density. By way of explanation, a probability distribution of a data set specifies the prior probabilities of the classes.

In this particular work, each context consists of all mixtures of the probability distributions functions (PDF) of the classes, taken individually, where the PDF for each class is constant across all mixtures. In other words, given a fixed PDF *for each class*, all variations of class proportion are considered as the same context. Formally, and in accordance to our previous notation, in this application a context defines $P(X|Y)$. It does not specify $P(Y|X)$ since this probability is impacted by changes in the proportions of classes, especially when there is class overlap. In this work, we also restrict the context identification task to binary classification problems.

In the next sections, we describe each one of our proposals.

4.3.1 SMR-HDy

Our first algorithm, Single Most Relevant HDy (SMR-HDy), is a simple approach to the context identification task that serves as a base for the second algorithm, which we describe later.

Consider the set $\mathcal{C} = \{C_1, \dots, C_z\}$ of contexts. For each context $C_i \in \mathcal{C}$, we have available training sets U_i and validation sets V_i . Each data point x_i in such sets is associated

with a class label $y(x_i) \in \{c_+, c_-\}$. Let $V_i^+ \subseteq V_i$ be the subset of V_i that contains all data points associated with the positive class c_+ . Consider V_i^- the analogous for the negative data points.

From the training sets U_i , we induce the scorers h_i^s . The algorithm proceeds by computing normalized (unit sum) histograms H_i^+ and H_i^- of the scores obtained by h_i^s on the validation sets V_i^+ and V_i^- , respectively.

Finally, given an unlabeled test set U , we compute the normalized histogram H_i^U of the scores obtained by the scorer h_i^s on the test set U . We then consider the most probable context for U as:

$$\hat{C} = \text{SMR-HDy}(U) = C_{f^s(U)}$$

$$\text{where } f^s(U) = \arg \min_{1 \leq i \leq |\mathcal{C}|} \text{HD}(\alpha_i H_i^+ + (1 - \alpha_i) H_i^-, H_i^U)$$

$$\text{and } \alpha_i = \text{HDy}(H_i^+, H_i^-, H_i^U)$$

In other words, $\hat{C}$ is the context that minimizes the divergence between the distributions of scores obtained by the scorer $h_{f^s(U)}^s$ on the validation set $V_{f^s(U)}$ and on the unlabeled sample U . This minimization is found for the particular class proportion $\alpha_{f^s(U)}$.

The rationale behind the algorithm is that HDy minimizes the divergence between a test histogram and a linear interpolation between the two training histograms within a context. In addition to the class proportion estimate, we obtain a measurement for the divergence that can be used to compare different contexts (the Hellinger Distance itself, which is usually disregarded by HDy).

We note that, in SMR-HDy, we model the expectation of behavior for scores produced by each scorer on data points that belong only to its corresponding context. This is a simple approach, and we are in fact discarding useful information: the expected behavior of the scores obtained by scorer h_i^s over the validation sets V_j when $i \neq j$. We overcome this limitation in our second algorithm, presented in the next section.

4.3.2 XO-HDy

Our second proposal, Crossed Opinions HDy (XO-HDy), takes into account how the scores for data points from one context are expected to behave when obtained by a scorer trained on a different context.

Consider $H_{i,j}^+$ the normalized histogram of scores obtained by the scorer h_i^s on the validation set V_j^+ . $H_{i,j}^-$ is the analogous for V_j^- . Also, consider $\alpha_{i,j}$ as follows:

$$\alpha_{i,j} = \text{HDy}(H_{i,j}^+, H_{i,j}^-, H_i^U)$$

We then consider the most probable context for U as:

$$\hat{C} = \text{XO-HDy}(U) = C_{f^x(U)}$$

$$\text{where } f^x(U) = \arg \min_{1 \leq j \leq |\mathcal{C}|} \frac{1}{|\mathcal{C}|} \sum_{i=1}^{|\mathcal{C}|} \text{HD} \left(\alpha_{i,j} H_{i,j}^+ + (1 - \alpha_{i,j}) H_{i,j}^-, H_i^U \right)$$

Figure 32 provides a visual illustration of the sources of data inside the computation of each Hellinger Distance in XO-HDy.

Figure 32 – *Opinion* from model h_i^s regarding context j . If scores obtained by h_i^s on the test data presents similar behavior to that of any mixture between positive and negative scores from h_i^s applied on V_j , then h_i^s votes in favor of the actual context being j .

4.3.3 Classification adjustment

A byproduct of our method is an estimation of the positive class ratio $\hat{p}$, which is either $\hat{p} = f^s(U)$ in the case of SMR-HDy, or $\hat{p} = f^x(U)$ in the case of XO-HDy. However, we note that once our method has identified the context, we can use any quantification algorithm in conjunction with the appropriate classifier, instead of HDy.

After the context $\hat{C}_i$ is inferred, we can readjust the classifier induced upon L_i with a new threshold and reclassify the events in U . Precisely, we expect $(100 \times \hat{p})\%$ of the examples to belong to the positive class. Therefore, we can set the classification threshold as the $(1 - \hat{\alpha})$ -quantile of the obtained scores; that is, we consider that the $(100 \times \hat{p})\%$ of the data points that obtained highest scores as positive.

4.3.4 Computational requirements

We can split the computational cost into two parts: training and deployment. The training phase is performed once, and the deployment phase is recurrent for every test data chunk.

Consider h_{tr}^s the cost in time to train a scorer, h_{mem}^s the cost in memory to store a scoring model, and h_{cl}^s the cost in time to issue a score for one data point. Notice that h_{tr}^s , h_{mem}^s , and h_{cl}^s depend on the scoring algorithm adopted. Additionally, consider $|V|$ the size of the validation set for each context and b the number of histogram bins for the Hellinger Distance.

The cost in time to train the SMR-HDy is $O(|\mathcal{C}|h_{\text{tr}}^s + |\mathcal{C}|h_{\text{cl}}^s|V|)$, while the cost to keep the SMR-HDy in memory after the training phase is $O(|\mathcal{C}|h_{\text{mem}}^s + |C|b)$. The training time cost of XO-HDy is $O(|\mathcal{C}|h_{\text{tr}}^s + |\mathcal{C}|^2h_{\text{cl}}^s|V|)$, while its post-training memory cost is $O(|\mathcal{C}|h_{\text{mem}}^s + |\mathcal{C}|^2b)$.

From a deployment perspective, the cost in time to use both SMR-HDy and XO-HDy depends on which search strategy we use for minimizing $\text{HD}(A + xB, C)$ inside the HDy algorithm. Although the original authors of HDy reckon that many strategies can be applied, they suggest using linear search. In this case, the domain of the function $([0, 1])$ is discretized into ρ parts, and the cost in time of estimating its minimum is $O(\rho b)$. On the other hand, we advocate applying Ternary Search, even though the function to be minimized is not unimodal. Our experimental results show that the results are essentially the same as the ones obtained by a linear search, in spite of substantially lower computational cost: $O(bk_\epsilon)$, where k_ϵ grows logarithmically according to the inverse of the error ϵ , which is a parameter of the Ternary Search. For instance, $k_\epsilon = 24$ when $\epsilon = 10^{-4}$, and $k_\epsilon = 30$ when $\epsilon = 10^{-5}$.

Consider $|U|$ the number of data points in the test set U . The time complexity of running SMR-HDy depends on the time required to compute the test histograms H_i^U ($O(|\mathcal{C}|h_{\text{cl}}^s|U|)$), and the time required to identify the current context ($O(|\mathcal{C}|bk_\epsilon)$). In total, it is $O(|\mathcal{C}|h_{\text{cl}}^s|U| + |\mathcal{C}|bk_\epsilon)$.

Similarly, the cost of running XO-HDy comprises the time required to compute the histograms ($O(|\mathcal{C}|h_{\text{cl}}^s|U|)$), and the time required to identify the current context ($O(|\mathcal{C}|^2bk_\epsilon)$). In total, it is $O(|\mathcal{C}|h_{\text{cl}}^s|U| + |\mathcal{C}|^2bk_\epsilon)$.

We note that if the test set U comprises a sliding window in a data stream, every time we observe a data point, the histograms are updated in $|\mathcal{C}|h_{\text{cl}}^s$. In this case, it is unnecessary to recompute the histograms entirely. We note, however, that employing a sliding window increases the memory cost by $O(|U||\mathcal{C}|)$, given the need for a queue to remove old values from the histograms in the correct order.

Additionally, it is unnecessary to keep all $|U|$ data points from the test set U in memory, simultaneously. Each observation can be processed and promptly discarded, one at a time.

4.4 Experimental setup

In this section, we describe the experimental setup and datasets adopted to evaluate our methods.

4.4.1 Algorithms and Evaluation

Although the proposed algorithms aim to identify the correct context of a sample of unlabeled data points, the rationale behind this objective is the existence of some underlying task that could potentially achieve better performance when considering the context information. Therefore, we not only measure the context identification accuracy, *i.e.*, how many times our methods predict the correct context, but also the performance for the classification and quantification tasks when relying on such predictions.

We compare our algorithms' performances with two topline, two baselines, and the Kolmogorov-Smirnov (KS) test. Their reasoning is as follows:

- $\mathcal{T}_1$ (Topline 1): this algorithm always predicts the correct context. We expect it to perform as well or better than our proposals. We use this algorithm to obtain an upper limit for the underlying tasks (classification and quantification) when we consider separate contexts and identify the correct one among them flawlessly;
- $\mathcal{T}_2$ (Topline 2): in this case, the correct context is known and is used as an additional feature to describe each example. We induce only one classifier using all available training data, after including this additional feature. We believe that this is the way most practitioners would model their problems, in case they had access to the variables responsible for defining the contexts. If the first topline or the proposed approaches surpass this topline, we have evidence that it is better to induce individual models for each context instead of considering a single model with a context attribute in the data point;
- $\mathcal{B}_1$ (Baseline 1): similarly to the second topline, this approach only induces one classifier with all available training data. However, this baseline disregards context information. If this baseline performs as well as the topline, there is an indication that there is no concept drift (or dataset shift) across different contexts. Otherwise we have evidence that the data distribution changes across contexts;
- $\mathcal{B}_2$ (Baseline 2): this baseline guesses the context uniformly at random. Since in our experiments all contexts occur with equal chance, surpassing this baseline indicates that our proposals actually learned how to predict the context.
- KS (Kolmogorov-Smirnov): 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; KORYCKI; KRAWCZYK, 2019). We compare the distributions of the scores output by each classifier and choose the one with the lowest statistic. As this method does not estimate the positive class ratio, we apply HDy after choosing the context.

Section 4.4.2 describes the six real datasets used in our experiments. We uniformly and randomly split each dataset into two disjoint halves. The first half is reserved for training and validation, while the second is used for testing. As all algorithms compared depend only on the scores generated by a model, we adopted 10-fold cross-validation to obtain the scores using the first half of the dataset. After obtaining all validation scores, we induced the scorers in the first half (one for each context, plus one for $\mathcal{T}_2$ and one for $\mathcal{B}_1$) and obtained scores for all events in the second half.

For each context, in each dataset, we varied the positive class proportion from 0% to 100%, with increments of 1%. For each positive ratio, we created 10 random samples of W events. No event appears more than once in each sample, though they can appear on multiple samples. Such samples were used to test and measure the performance of the competing approaches. All-in-all, we evaluated $1000 \times |\mathcal{C}|$ samples for each dataset, in which $|\mathcal{C}|$ is number of contexts.

We also report the quantification results obtained for Adjusted Classify And Count (ACC) coupled with all competing approaches. For instance, ACC with XO-HDy uses the classifications from the classifier trained on the context predicted by XO-HDy, and ACC with $\mathcal{B}_\epsilon$ uses the classifications from a single classifier without context information.

We measured the performance of the approaches regarding context identification accuracy (only for SMR-HDy and XO-HDy), the classification accuracy after choosing the context, and mean absolute error for the quantification task.

We merged all test samples into a single stream to analyze our approach in data streams scenarios. In this case, each context appears in the stream in the form of 10 non-consecutive sequences of instances. Each sequence is composed of the concatenation of 100 randomly positioned samples of size W , each sample having a single positive class ratio within this sequence. We compare XO-HDy against MINAS (FARIA *et al.*, 2016b) according to non-adjusted classification accuracy. We used the authors' freely available implementation of MINAS with default settings, except for the threshold for Novelty Detection. We raised this threshold so that MINAS detected no more than 50 novelties in the stream.

All classification models and scorers are Random Forests with 500 trees, using scikit-learn¹ with default parameters.

¹ scikit-learn v0.19.0 available at <http://scikit-learn.org/>

Since HDy is still a relatively new algorithm, there is no general consensus regarding an optimal number b of bins for the histograms. The original authors of HDy did not suggest any specific value, though they ran all experiments with varying sizes from 10 to 100 with increments of 10 and reported the median result. We note that choosing b , although a non-trivial task, is not devoid of useful insights. Histograms with large values of b are negatively affected by two aspects. First, if b is not large enough, such histograms can become too sparse, each bin can have too low weight and, ultimately, the Hellinger Distance can face the curse of dimensionality. Second, a large number of bins has the strong assumption of high precision for the scores. On the other hand, if b is too small, the Hellinger Distance can be unable to differentiate distributions. For the sake of simplicity, we opted for using $b = 11$ across all experiments, meaning that we allow for a precision of 0.1 for the scores (from 0 to 1.0). Maletzke *et al.* (2019) perform a more in-depth study on this matter.

Finally, to verify if we can safely adopt Ternary Search inside HDy, we simulated the same conditions as described above, and measured the difference between the positive class ratios found by HDy with linear search and with Ternary Search. For the linear search, we split the interval $[0, 1]$ into 1000 uniformly distributed parts. For the Ternary Search, we adopted an error of 10^{-5} .

For the sake of reproducibility and further usage of our ideas, we make available all the codes, datasets, and detailed results at our supplementary material repository².

The next section describes each dataset used in our experimental setup.

4.4.2 Datasets Description

This section describes each dataset used to benchmark our proposals. As the datasets have a limited number of data points, we chose W (size of the test sample) so that there would be enough examples to vary the probability of the positive class from 0 to 1, and still have variability across samples in the extremes of such a range. The datasets are:

Aedes-Culex The dataset *Aedes aegypti-Culex quinquefasciatus* (REIS; MALETZKE; BATISTA, 2018) contains 24,000 events described by features that register the passage of female *Aedes aegypti* and female *Culex quinquefasciatus* mosquitoes in front of a sensor. Besides wing beat frequency, there are other 25 numerical features obtained from the signal power spectrum. Six ranges of temperature define the contexts, and the objective is to distinguish between the two species. $W = 300$;

² <<https://github.com/denismr/Classification-and-Counting-with-Recurrent-Contexts>>

Aedes-Sex The dataset *Aedes aegypti*-sex (REIS; MALETZKE; BATISTA, 2018), also containing 24,000 entries, is similar to the previous one. However, we want to discriminate between male and female *Aedes aegypti* mosquitoes. $W = 300$;

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 context, and the task is to predict the sex of the speaker. $W = 200$;

CMC CMC (LICHMAN, 2013) is a dataset from a survey on contraceptive prevalence at Indonesia. It contains 1,473 entries described by eight features. The task is to predict whether a contraceptive is in use or not, and the concept is given by two ranges of age. $W = 100$. As MINAS requires purely numerical datasets, we excluded this dataset in its evaluation;

Handwritten-QG QG is a subset of the dataset Handwritten (REIS; MALETZKE; BATISTA, 2018), which contains 4,010 entries described by 63 numerical features regarding the handwritten letters g , and q . The context is given by the who wrote the letter (among 10 people), and the objective is to predict the letter. $W = 80$;

Wine Wine Quality (CORTEZ *et al.*, 2009; LICHMAN, 2013) contains 6,498 entries described by 11 features. The task is to differentiate between two ranges of quality of wines. The context is given by the type of the wine (red or white). $W = 300$.

4.5 Experimental evaluation

In this section, we present and discuss our experimental results.

We open this section by displaying evidence that supports using ternary search inside the HDy algorithm. Table 6 presents the average absolute differences between the positive ratios found when applying discretized linear search and ternary search inside HDy. The most significant average difference occurs for dataset CMC, and this value represents a single instance every 2,645 instances. In other words, linear search and ternary search would diverge by one single data point in a data sample containing 2,645 observations. We emphasize that such differences were obtained when comparing $\arg \min_{\alpha} \text{HD}(\alpha)$. After mapping the values found by each search method to $\text{HD}(\alpha)$, ternary search produces smaller distances in most cases (column $< \text{HD}$).

We also assess the accuracy of the context identification. Table 7 presents the results for mean accuracy for all class distributions. Our methods performed consistently above chance level ($\mathcal{B}_2$), and XO-HDy consistently outperformed both SMR-HDy and KS. We note that XO-HDy achieved perfect context identification in three out of six datasets, almost perfect in a fourth, and the lowest identification accuracy was 88.73%.

Table 6 – Average absolute difference ($\overline{\Delta x}$) between the positive ratios found by discretized linear search and by ternary search, in the HDy algorithm. Column # represents the number of trials and column s presents the standard deviation. The last column presents the proportion of cases in which the ternary search led to a lower Hellinger Distance.

Dataset	#	$\overline{\Delta x}$	s	< HD
Aedes-Culex	387840	1.78×10^{-4}	1.65×10^{-4}	70.49%
Aedes-Sex	387840	2.47×10^{-4}	1.46×10^{-4}	98.45%
Arabic-Digit	1454400	2.44×10^{-4}	1.49×10^{-4}	97.21%
CMC	32320	1.75×10^{-4}	1.65×10^{-4}	68.95%
Handwritten-QG	1454400	3.78×10^{-4}	6.50×10^{-3}	84.64%
Wine	32320	1.89×10^{-4}	1.64×10^{-4}	75.17%

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

Table 7 – Mean Context identification accuracy (%) comparison between a uniformly random choice ($\mathcal{B}_2$), Kolmogorov-Smirnov test (KS), and our two proposals.

Dataset	$\mathcal{B}_2$	KS	SMR-HDy	XO-HDy
Aedes-Culex	16.95	46.07	61.57	88.73
Aedes-Sex	16.95	39.65	65.66	99.77
Arabic-Digit	10.50	66.87	99.59	100.00
CMC	51.34	80.59	71.19	94.21
Handwritten-QG	10.08	80.78	88.59	100.00
Wine	48.71	90.00	100.00	100.00

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

SMR outperformed KS in all dataset but CMC. KS performs well mainly when the positive class ratio is similar to that in the training set, and poorly otherwise. Figure 33 illustrates the context identification accuracy for each positive class proportion on dataset Aedes-Culex. Other datasets have similar performance (shown in the supplemental material).

In the remainder of this section, we present the results for the underlying tasks, that is, quantification and classification.

4.5.1 Quantification

Table 8 presents the mean absolute error (MAE) between the predicted and actual positive ratios obtained with HDy. The mean values are computed across all positive distributions.

For the quantification tasks, XO-HDy achieved the best performance among all methods. The fact that XO-HDy performed better than $\mathcal{B}_1$ highlights the importance of differentiating contexts. Similarly, the fact it also outperformed $\mathcal{B}_2$ indicates the relevance of using the *correct* context. In fact, $\mathcal{B}_2$ performed consistently worse than $\mathcal{B}_1$. This

Figure 33 – Context identification accuracy for all positive class proportions for dataset Aedes-Culex.

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

Table 8 – HDy absolute quantification error (%) averaged over all positive proportions. Standard deviation in parentheses.

Data	$\mathcal{B}_1$	$\mathcal{B}_2$	KS	SMR	XO	$\mathcal{T}_1$	$\mathcal{T}_2$
C	27.04 (20.29)	28.23 (26.32)	19.65 (20.97)	17.87 (24.96)	6.63 (14.66)	2.20 (1.78)	5.95 (4.18)
S	4.14 (4.64)	9.38 (16.01)	7.36 (16.13)	0.62 (0.75)	0.44 (0.40)	0.44 (0.40)	1.00 (1.15)
D	1.13 (0.92)	19.96 (19.36)	6.19 (12.00)	0.71 (0.64)	0.70 (0.63)	0.70 (0.63)	0.97 (0.80)
M	25.24 (14.98)	29.15 (26.24)	23.10 (22.90)	27.60 (28.70)	16.71 (18.21)	12.73 (10.77)	10.69 (8.60)
Q	1.96 (3.17)	22.87 (25.34)	1.47 (5.71)	0.82 (0.70)	0.80 (0.66)	0.80 (0.66)	1.90 (2.93)
W	13.04 (10.65)	26.14 (28.83)	10.08 (18.76)	3.94 (2.66)	3.94 (2.66)	3.94 (2.66)	13.28 (11.50)

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

further supports the previous point, and more importantly, it also clarifies that it is better to ignore context information rather than using it if the context information is unreliable.

As XO-HDy obtained perfect context identification in three datasets, and almost perfect in a fourth dataset, it performed as well as the topline $\mathcal{T}_1$ in such cases. However, XO-HDy performed worse than $\mathcal{T}_1$ for the other two datasets. Although a paired t-test between XO-HDy and $\mathcal{T}_1$ results in a p -value of 17.55%, we believe that this p -value would decrease as we increase our sample size, as we expect $\mathcal{T}_1$ to be an upper bound of the performance obtainable by XO-HDy. A paired t-test between XO-HDy and $\mathcal{T}_2$ resulted in a p -value of 72.14%, which does not indicate a statistically significant difference between the methods. An important observation, though, is that XO-HDy only underperformed $\mathcal{T}_2$ when it did not achieve almost perfect context identification. Finally, $\mathcal{T}_1$ only underperformed $\mathcal{T}_2$ in one dataset. This indicates that in most of our cases, separating the contexts is a better approach than including it as a feature.

In general, our results suggest that in many cases, we can confidently use XO-HDy in quantification tasks without context information, and expect the same results as

those obtained if context information were available. In other cases, although XO-HDy performs worse than when context information is available, it still outperforms approaches that simply ignore contexts or randomly guess the current context (as done by $\mathcal{B}_1$ and $\mathcal{B}_2$).

Table 9 presents the quantification errors when using Adjusted Classify and Count (ACC) instead of HDy. We did not observe significant differences between HDy and ACC. Indeed, ACC outperforms HDy in specific datasets; however, it still benefits from context identification the same way HDy does and relies on HDy for this task.

Table 9 – ACC absolute quantification error (%) averaged over all positive proportions.

Data	$\mathcal{B}_1$	$\mathcal{B}_2$	KS	SMR	XO	$\mathcal{T}_1$	$\mathcal{T}_2$
C	23.09 (18.25)	28.53 (26.16)	19.54 (20.57)	18.19 (24.83)	7.14 (14.59)	2.84 (2.21)	5.84 (3.72)
S	3.24 (4.67)	8.44 (14.99)	7.46 (15.00)	0.65 (0.72)	0.52 (0.45)	0.52 (0.45)	0.84 (0.64)
D	1.00 (0.69)	12.54 (14.01)	6.51 (11.93)	0.59 (0.53)	0.59 (0.53)	0.59 (0.53)	1.00 (0.69)
M	22.55 (17.43)	31.12 (27.32)	26.53 (26.30)	29.02 (29.19)	19.10 (20.51)	14.88 (13.72)	16.20 (12.10)
Q	1.18 (1.67)	11.15 (20.27)	1.04 (4.96)	0.61 (0.49)	0.60 (0.43)	0.60 (0.43)	1.18 (1.67)
W	13.56 (10.70)	18.95 (24.60)	7.37 (6.24)	5.74 (3.53)	5.74 (3.53)	5.74 (3.53)	13.22 (10.45)

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

4.5.2 Classification

Tables 10 and 11 present non-adjusted and adjusted classification accuracies, respectively. The observations regarding $\mathcal{B}_2$ are the same as the ones made for quantification. All p -values reported are according to the paired t-test.

We cannot statistically infer that XO-HDy outperforms $\mathcal{B}_1$ (p -value = 32.29%). However, when we compare the adjusted XO-HDy against the best results between non-adjusted and adjusted $\mathcal{B}_1$, XO-HDy outperforms $\mathcal{B}_1$ in every single dataset. Comparing adjusted XO-HDy against the best between adjusted and non-adjusted $\mathcal{T}_1$ and $\mathcal{T}_2$ lead to the same conclusions than the ones obtained from the quantification results.

Table 10 – Classification accuracy (%) averaged for all positive class proportions.

Data	$\mathcal{B}_1$	$\mathcal{B}_2$	KS	SMR	XO	$\mathcal{T}_1$	$\mathcal{T}_2$
C	76.84 (14.42)	72.32 (19.87)	74.40 (13.66)	76.37 (15.66)	82.02 (9.77)	84.01 (4.96)	83.94 (5.39)
S	96.14 (4.87)	89.25 (16.69)	91.66 (15.21)	98.58 (1.14)	98.79 (1.02)	98.79 (1.02)	98.51 (1.11)
D	98.96 (1.01)	84.54 (14.22)	93.30 (12.07)	99.48 (0.64)	99.48 (0.64)	99.48 (0.64)	98.89 (0.98)
M	66.07 (6.74)	62.96 (10.66)	61.72 (9.94)	61.71 (9.63)	64.03 (7.74)	65.45 (5.20)	66.51 (5.47)
Q	99.20 (1.82)	88.04 (20.97)	99.26 (5.03)	99.71 (0.62)	99.71 (0.57)	99.71 (0.57)	99.20 (1.82)
W	78.30 (5.51)	69.42 (16.37)	77.75 (6.12)	78.18 (5.36)	78.18 (5.36)	78.18 (5.36)	78.62 (5.43)

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

Figure 34 illustrates the average accuracy obtained by both adjusted and non-adjusted XO-HDy in dataset Aedes-Culex, for every positive class ratio tested. From this figure, we can see that a possible approach is to adjust a classifier only when the quantification is within specific ranges of the positive class proportion, and improve classification

Table 11 – Mean classification accuracy (%) for all positive class proportions with decision threshold adjusted according to quantification results in Table 8.

Data	$\mathcal{B}_1$	$\mathcal{B}_2$	KS	SMR	XO	$\mathcal{T}_1$	$\mathcal{T}_2$
C	70.87 (18.98)	68.95 (24.28)	73.51 (16.85)	75.97 (22.02)	84.70 (13.54)	88.07 (5.83)	87.26 (5.83)
S	95.50 (4.66)	88.77 (16.54)	91.81 (16.18)	98.68 (1.15)	98.90 (0.94)	98.90 (0.94)	98.53 (1.37)
D	98.73 (0.98)	78.80 (19.18)	93.68 (12.01)	99.23 (0.68)	99.24 (0.67)	99.24 (0.67)	98.88 (0.89)
M	68.15 (15.52)	62.92 (22.66)	64.14 (19.14)	61.79 (23.70)	68.70 (16.14)	72.32 (11.99)	73.31 (11.80)
Q	98.29 (3.15)	76.38 (25.71)	98.68 (5.77)	99.41 (0.85)	99.44 (0.83)	99.44 (0.83)	98.37 (2.94)
W	80.83 (9.37)	68.45 (25.39)	79.25 (16.38)	85.10 (6.18)	85.09 (6.18)	85.10 (6.18)	80.48 (9.76)

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

accuracy even further. This pattern was common across all data sets. We believe that, while adjustment increases performance for samples that have class proportions different than the one found in the training set (which in our case is balanced), it may have the opposite effect otherwise.

Figure 34 – Comparison between adjusted and non-adjusted classification performance across different class proportions. Data obtained with method XO-HDy on dataset Aedes-Culex.

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

4.5.3 Data Stream Classification

Table 12 presents the accuracies obtained by XO-HDy and MINAS (FARIA *et al.*, 2016b) in a streaming setting. Although we fed both with the same training data, MINAS was unable to discern the contexts and, judging by its inferior performance compared to $\mathcal{B}_1$, its forgetting mechanism was harmful. XO-HDy consistently outperformed MINAS. Our results suggest that when contexts are known, switching between them is better than adapting the models after tracking incremental drifts.

Table 12 – Comparison of classification accuracy (%) between MINAS (FARIA *et al.*, 2016b) and XO-HDy in data stream setting.

Dataset	MINAS	XO-HDy
Aedes-Culex	58.30	82.91
Aedes-Sex	91.91	98.68
Arabic-Digit	56.97	99.43
Handwritten-QG	74.57	99.61
Wine	48.20	77.49

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

4.6 Related additional contributions

In this section, we discuss additional contributions that we made and relate to the contents of this chapter.

4.6.1 Sample ORD

Maletzke *et al.* (2019) generalized HDy into DyS. The point of generalization is the replacement of Hellinger Distance: the dissimilarity function between distributions becomes a parameter of the method. We note that the same generalization is possible for SMR-HDy and XO-HDy.

The authors examined the behavior of several distance functions for histograms (amongst them the Hellinger Distance) and the the best choice of b for each one. All but one distance function based on histograms were severely affected by varying number of bins. ORD (CHA; SRIHARI, 2002), on the other hand, was mostly unimpaired. The difference of ORD to other distance functions is that it takes into account the relation of unaligned bins, as illustrated in Figure 35.

The objective of ORD is to find the least number of movements needed in order to transform one histogram into the other, where one movement is the transference of a unit from a bin to one of its immediate neighbor bins. There is always a possible transformation if the histograms are normalized so that the same number of units. ORD is a univariate case of the *minimum difference of pair assignment* (MDPA) of two distributions, which in turn is a special case of the Earth Mover’s Distance (EMD) (RUBNER; TOMASI; GUIBAS, 1998).

The proposers of ORD introduced 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. Algorithm 1 describes such an algorithm.

We introduce Sample ORD (SORD), a variation of ORD that compares sample of univariate data points instead of histograms. SORD can be interpreted as a special case

Figure 35 – Difference between ORD and other dissimilarity functions for histograms. Consider a histogram that count integer numbers and has three bins. ORD considers the distance between unaligned units.

Source: Elaborated by the author.

Algorithm 1 – Ordinal Distance

```

1: procedure ORD( $S, Z$ )                                ▷ Distance between  $S$  and  $Z$ 
2:    $\text{diffsum} \leftarrow 0$ 
3:    $\text{total\_cost} \leftarrow 0$ 
4:   for  $i \leftarrow 1$  to  $\text{length}(S)$  do
5:      $\text{diffsum} \leftarrow \text{diffsum} + (S[i] - Z[i])$ 
6:      $\text{total\_cost} \leftarrow \text{total\_cost} + |\text{diffsum}|$ 
7:   end for
8:   return  $\text{total\_cost}$ 
9: end procedure

```

of ORD where the number of bins is infinity. Therefore, SORD is the minimum cost to transform a sample into another one.

SORD, as a dissimilarity function between two samples S and Z , weights every data point x 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 $f_{i,j}$, where $0 \leq f_{i,j} \leq 1$, 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 described by the following optimization problem.

$$\begin{aligned}
& \underset{f_{i,j} \forall i,j}{\text{minimize}} \sum_i^{|S|} \sum_j^{|Z|} c(i, j) \\
& \text{subject to } \sum_j^{|Z|} f_{i,j} = 1 \quad \forall i \\
& w(Z_j)Z_j = w(S_i)S_i \sum_i^{|S|} f_{i,j} \quad \forall j
\end{aligned} \tag{4.1}$$

For the purpose of quantification, one of the samples compared by SORD is a mixture of two other samples: one that contains scores from positive observations and another with scores from negative 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 2](#) 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 2 – Sample ORD

```

1: procedure SORD( $S^+, S^-, Z, \alpha$ )  $\triangleright$  Dissimilarity between  $\alpha S^+ + (1 - \alpha)S^-$  and  $Z$ 
2:    $w'(x) := \begin{cases} \alpha |S^+|^{-1}, & x \in S^+ \\ (1 - \alpha) |S^-|^{-1}, & x \in S^- \\ -|Z|^{-1}, & x \in Z \end{cases}$ 
3:    $v \leftarrow$  sorted array with  $\forall x \in S^+ \cup S^- \cup Z$ 
4:    $\text{acc} \leftarrow w'(v[1])$ 
5:    $\text{total\_cost} \leftarrow 0$ 
6:   for  $i \leftarrow 2$  to  $\text{length}(v)$  do
7:      $\delta \leftarrow v[i] - v[i-1]$ 
8:      $\text{total\_cost} \leftarrow \text{total\_cost} + |\delta \times \text{acc}|$ 
9:      $\text{acc} \leftarrow \text{acc} + w'(v[i])$ 
10:  end for
11:  return  $\text{total\_cost}$ 
12: end procedure

```

[Maletzke et al. \(2019\)](#) demonstrated SORD to be the best dissimilarity function for quantification using DyS, when the number of bins is not tuned for other distances. Moreover, even when this parameter is in fact tuned, the performance is only marginally superior to that of SORD.

We evaluated the replacement of HD with SORD in the SMR and XO algorithms for context identification, and show our results in Table 13. The major difference is for SMR, where, in all occasions SORD gave the algorithm an advantage, the difference surpassed 10%. On the other hand, on the occasions where SMR displayed better performance with HD, the difference was inferior to 1%.

Table 13 – Mean Context identification accuracy (%) for SMR and XO algorithms with HD and SORD.

Dataset	HD		SORD	
	SMR	XO	SMR	XO
Aedes-Culex	61.57	88.73	71.58	87.06
Aedes-Sex	65.66	99.77	79.74	99.98
Arabic-Digit	99.59	100	99.11	100
CMC	71.19	94.21	68.02	97.87
Handwritten-QG	88.59	100	100	100
Wine	100	100	100	100

Source: Research data.

While Maletzke *et al.* (2019) already demonstrated the superiority of SORD for quantification, the task of classification is impacted solely by the context identification accuracy. As XO was not severely improved by SORD, the underlying classification task was not statistically different. For this reason, we omit further results in this analysis. In our supplemental material website, we include such results in addition to the use of other distances: Jensen Difference, Mixable Kolmogorov-Smirnov (discussed next), L-1, L-2, L- ∞ , ORD, Probabilistic Symmetric, Taneja and Topsøe.

4.6.2 Mixable Kolmogorov-Smirnov

We also introduce an adaptation of the Kolmogorov-Smirnov statistic called Mixable Kolmogorov-Smirnov (MKS) statistic. In MKS, we compare two discrete empirical distributions and account for the first distribution being a weighted mixture of two distributions.

Equation (4.2) formalizes MKS, where S^+ and S^- are the two samples that are 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.

$$D_{MKS}(S^+, S^-, \alpha, Z) = \sup_x |\alpha F_{S^+}(x) + (1 - \alpha) F_{S^-}(x) - F_Z(x)| \quad (4.2)$$

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

Unfortunately, MKS enjoys a poor performance in comparison to other dissimilarity functions (MALETZKE *et al.*, 2019).

4.6.3 Model for Score Simulation

One limitation of both SMR-HDy and XO-HDy is the necessity of previously knowledge about all possible contexts. In real-world applications, this scenario may be unfeasible. To recognize unexpected contexts, one can set a threshold for the dissimilarity between the closest context and the test data: values above the threshold would indicate novelty and/or unknown context. Maletzke *et al.* (2018) provide pointers on how to define such a threshold based on empirical percentiles.

However, the detection of novel contexts does not completely solve the problem. In Section 4.1.2, we criticized methods that only detect changes in data, since we are assuming true labels will be always unavailable for test samples.

Maletzke, A. (MALETZKE, 2019) developed DySyn, a quantification method that can perform well even when the distribution of data changed, given that the changes are not extreme. This method can make up for scenarios where we have unexpected contexts.

DySyn is an extension of the ideas in DyS and HDy, and it contains two major differences. First, DySyn compares the distribution of scores obtained from test data against the distribution of synthetic scores, rather than scores of training data. The reasoning is that a precise estimation of the distribution of training scores is not needed for the purpose of quantification, and a roughly defined *shape* of the expected distribution suffices.

The second difference is that the method varies a parameter $\mathbf{m}$ of the synthetic scores that defines how the classes are overlapping. The rationale is the changes in the data distribution, which can make the underlying classification task easier or harder, influences the separability of scores of positive and negative data points. The search task of DySyn is depicted in Figure 36.

DySyn is based on our proposed **Model for Score Simulation** (MoSS). MoSS depends on one parameter, the *merging* factor $\mathbf{m} \in [0, 1]$, which represents the difficulty of the underling classification task. When $\mathbf{m} = 0$, 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 $\mathbf{m} = 1$, all scores for both classes are uniformly distributed within the interval $[0, 1]$, turning it impossible to distinguish the 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.

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

Figure 36 – 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 the author.

where

$$\begin{aligned} \text{syn}(\oplus, n, \mathbf{m}) &= \bigcup_{i=1}^n \{X_i^{\mathbf{m}}\} \quad , \quad X_i \sim \text{Uniform}(0, 1) \\ \text{syn}(\ominus, n, \mathbf{m}) &= \bigcup_{i=1}^n \{1 - X_i^{\mathbf{m}}\} \quad , \quad X_i \sim \text{Uniform}(0, 1) \end{aligned} \quad (4.3)$$

A synthetic score in **MoSS** is a non-linear map from a uniformly distributed random variable in the range $[0, 1]$ on 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 unmapped values of u and v , as illustrated in [Figure 37](#) and described below:

$$\begin{aligned} \text{syn}_{\text{density}}(\oplus, u, v, \mathbf{m}) &= v^{1/\mathbf{m}} - u^{1/\mathbf{m}} \\ \text{syn}_{\text{density}}(\ominus, u, v, \mathbf{m}) &= (1 - u)^{1/\mathbf{m}} - (1 - v)^{1/\mathbf{m}} \\ \text{MoSS}_{\text{density}}(\alpha, u, v, \mathbf{m}) &= \\ &\quad \alpha \text{syn}_{\text{density}}(\oplus, u, v, \mathbf{m}) + (1 - \alpha) \text{syn}_{\text{density}}(\ominus, u, v, \mathbf{m}) \end{aligned} \quad (4.4)$$

[Equation \(4.4\)](#) is not defined for $\mathbf{m} = 0$, 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 calculated 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 \(4.4\)](#).

We can apply the same consideration regarding the base uniform distribution to

Figure 37 – Relation between density of MoSS scores and uniform distribution. Only positive scores are illustrated. Negative scores are analogous. The hatched regions have the same area.

Source: Elaborated by the author.

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:

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

Finally, we also can compute the area under the ROC curve (AUC) (FLACH, 2012) of **MoSS** scores without synthesizing them, similarly to what we previously did regarding their density. AUC is a widely used measure of the quality of scores. Scores that have higher quality better separate negative from positive data: positive scores are grouped together with higher values, while negative scores are grouped together with lower values. AUC is the area under the ROC curve, which is given by the relation between true positive rate (TPR) and false positive rate (FPR), as illustrated in Figure 38.

From the definition of AUC, we can mathematically express it as follows:

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR}) d\text{FPR} \quad (4.5)$$

Considering an arbitrary threshold t , the TPR obtained by classifying **MoSS** scores is given by the proportion of positive data points that have a score *greater* than t , in relation to the total number of positive data points, as illustrated in Figure 39b. Similarly, the FPR is given by the proportion of negative data points that have a scorer *greater* than t , in relation to the total number of negative data points, as illustrated in Figure 39a.

Figure 38 – ROC and AUC for MoSS scores ($m = 0.43$).

Source: Research data.

Figure 39 – TPR and FPR of MoSS.

Source: Research data.

Therefore, by using the our findings in [Equation \(4.4\)](#), we can calculate TPR and FPR for moss as follows:

$$\begin{aligned}
 \text{TPR}_{\text{MoSS}} &= \text{syn}_{\text{density}}(\oplus, t, 1, m) \\
 &= 1^{1/m} - t^{1/m} \\
 &= 1 - \sqrt[m]{t}
 \end{aligned} \tag{4.6}$$

$$\begin{aligned}
 \text{FPR}_{\text{MoSS}} &= \text{syn}_{\text{density}}(\ominus, t, 1, m) \\
 &= (1 - t)^{1/m} - (1 - 1)^{1/m} \\
 &= (1 - t)^{1/m}
 \end{aligned} \tag{4.7}$$

By isolating t in Equation (4.7), we have:

$$\text{FPR}_{\text{MoSS}}^m = (1 - t)^{m/m} \longrightarrow t = 1 - \text{FPR}_{\text{MoSS}}^m \quad (4.8)$$

After replacing t in Equation (4.6) with Equation (4.8), we obtain:

$$\text{TPR}_{\text{MoSS}}(\text{FPR}_{\text{MoSS}}) = 1 - \sqrt[m]{1 - \text{FPR}_{\text{MoSS}}^m} \quad (4.9)$$

Finally, using Equation (4.9) in Equation (4.5) gives us:

$$\begin{aligned} \text{AUC}_{\text{MoSS}}(m) &= \int_0^1 1 - \sqrt[m]{1 - x^m} dx \\ &= 1 - \frac{\Gamma(1 + \frac{1}{m})^2}{\Gamma(\frac{m+2}{m})} \end{aligned}$$

where Γ is the gamma function (DAVIS, 1959).

4.7 Discussion

Our proposed methods (SMR-HDy and XO-HDy) fill in a gap in literature by providing ways to reuse classification models when the number of possible concepts is known and limited. Additionally, they are capable of differentiating between prior probability shift and other aspects that make concepts different. In this regard, concepts are modeled independently of the proportion of classes. By assessing the current proportion, the methods can also improve classification performance by dynamically adjusting the decision threshold. For these reasons, in this chapter we fulfilled our first two contributions listed in Section 1.1.

SMR-HDy and XO-HDy have three significant limitations. The first is the fact that they are suitable only for binary classification problems. Extending our methods or developing new ones to multiclass settings is regarded as future work.

The second limitation is the assumption of purity of context, *i.e.*, adjacent instances belong to the same context. Our experimental evaluation reflects this assumption as, except when comparing our approaches against MINAS, we measure performance on batches that contain data from only one context at a time. Indeed, we expect more than one context to coexist within a same sliding window, in a data stream, during the transition between two consecutive contexts.

However, two aspects soften this last limitation. First, in the specific case of incremental drift, the period of coexistence can only be as long as the size of the sliding window.

We expect the window to contain much fewer instances than the number of observed examples associated with the active context. Therefore, this assumption is not likely to affect the measured performance significantly. Second, for some applications, it is reasonable to assume that consecutive contexts share enough similarities so that applying either model would lead to acceptable performance. In that sense, even during transitions, the mixture of contexts would not pose a practical issue to the performance of the underlying task.

The third and final limitation is the assumption that we previously know all contexts. This limitation is discussed in [Section 4.6.3](#). Future work can mitigate this limitation by setting a threshold to the dissimilarity measured between test data and its most similar known context. If the dissimilarity goes beyond such a threshold, we can consider that there is an unknown context. [Maletzke et al. \(2018\)](#) offer some suggestions on how to properly set such a threshold.

Although the methods presented thus far are appreciable contributions to the state-of-the-art in concept drift detection, and consequently are worth future improvements, they are quite astray from the requirements of our motivational application. Indeed, its most challenging aspect is the irredeemable fact that there are countless classes (insect species) and only a tiny fraction of them are bound to be analyzed prior to deployment. Many species that were unseen in laboratory will nonetheless be captured and analyzed by the trap. Their presence will undoubtedly deteriorate the performance of the model employed for either quantification or classification, regardless of the fact that we are only concerned about a very small number of species. Therefore, even developing a multiclass version of XO-HDy would not suffice. Neither would suffice to include all known negative into a single *negative* label, as explained in [Section 3.2.4](#).

We note that the main interest of our motivating application is gathering statistics to support decision making regarding vector control. Therefore, quantification is a more valuable task in contrast to classification. In this sense, if we cease the use of external sensors, we can better focus on quantification efforts rather than identifying context. For the quantification task, we present and describe methods that can overcome the unpredictability problem, in the next chapter.

ONE-CLASS QUANTIFICATION

The contents of this chapter are based on our published paper *One-Class Quantification* (REIS *et al.*, 2018a) and new research, which is included in a journal paper that is under production.

In several quantification applications, we are mostly interested in counting a single or a small set of class labels. A typical approach found in literature is to model these applications as binary quantification problems and use the positive class to designate the group of interest. For example, the positive class can indicate the *Aedes* or *Anopheles* mosquito genus – vectors of terrible diseases such as Zika fever and malaria, respectively –, or a specific defect that affects battery duration in mobile phones. On the other hand, the negative class is the universe, *i.e.*, a broad set that contains all categories except the positive one. In continuation to the previous examples, the negative label would be all other insect species in an area and other possible defects reported by customers.

The universe is a mixture of distributions, which we henceforth call sub-classes. Admittedly, characterizing the universe is not a trivial task. In addition to being typically represented by an exceedingly large number of sub-classes, for many applications, we may expect to eventually process observation points from distributions that were never examined before. Furthermore, such data points may not behave similarly to any previously studied category. Taking the task of counting insects that belong to a certain species as an example, different species may be observed at different proportions, depending on which region is under study and what are the environmental conditions. Additionally, the number of insect species is estimated to be between six and ten million (CHAPMAN; SIMPSON; DOUGLAS, 2013), making it impossible to characterize the negative class reliably. As a consequence, after the trap is deployed in the field, the quantification model will make observations of unpredictable behavior. Even if we are not interested in counting such observations, their presence can significantly degrade the quantification performance. Therefore, we can only consider the training data of the *positive class*, exclusively, to be

a representative sample of its corresponding population.

Recall that open-set methods that make use of available data for any negative sub-class have to assume at least some sort of correlation between training and test samples. Consequently, if the universe contains an exceedingly large number of sub-classes, the ones for which we have data might not be enough to reliably model the general behavior of the negative class. In such cases, the negative class remains, for practical purposes, unpredictable.

In addition to that, there are legitimate cases where it is impractical to collect samples with only negative observations. For example, suppose that we sell a product online and we can track the preferences of our customers via their social media profiles. Our customers can be used as positive training data for the task of identifying who might be interested in purchasing our product. On the other hand, gathering data for the negative class is not as trivial. If we randomly sample online social media profiles, the resulting set would contain both people who are uninterested in the product and potential customers. An explicit data gathering for the negative class could involve an online poll, which is time consuming and can still generate potentially biased data.

In a second example, suppose that we want to count the number of people that are infected with a disease in a population. Due to procedure costs, people may test for a disease only if they are suspicious of having it. In that case, while we can have a sample of people that were positively tested for such a disease, our data for people who were negatively tested may be severely lacking and biased. In such a case, a random sample of people would include both people who are not infected and people who are infected but were never diagnosed.

If we are interested in quantifying one single positive class and we are unable to have a reliable representation of the negative class alone, we may need to rely solely on positive training data to induce a quantification model. Any available data regarding the negative class can instead be used to evaluate such a model, although we emphasize that the evaluation is still limited since such data misrepresent the negative class.

To tackle the described scenario, in this chapter we introduce the first One-class Quantification (OCQ) ([REIS *et al.*, 2018a](#)) methods. Once induced using solely positive observations, training data is no longer required. Such quantifiers serve the purpose of measuring how many observations in any previously unseen test sample share the same distribution as the training data. The key aspect of OCQ methods is the lack of assumptions regarding the negative class. In other words, the negative class is expected to be composed of unpredictable distributions.

5.1 Related work

To the best of our knowledge, this is the first work to propose methods that fall within the strict definition of One-class Quantification, as defined in [Section 5.2](#). However, as discussed in [Section 5.2.1](#), methods of Positive and Unlabeled Prior Estimation (PU PE) produce the same practical results in terms of quantification when the negative class is unpredictable. Therefore, comparisons between our proposals and the state-of-the-art in PU PE are due. Additionally, as the proposed OCQ methods rely on One-class Scorers, we also review the most commonly used and easily available options.

5.1.1 Positive and Unlabeled Prior Estimation

In this section, we review and describe six of the more prominent methods in PU PE literature, highlighting key aspects of their rationale and implications in practical use. We do our best to simplify the rationale behind each method and offer a more intuitive and approachable explanation that unveils the uniqueness of each algorithm. Four of them are later used in our experimental evaluation.

5.1.1.1 Elkan

To the best of our knowledge, [Elkan and Noto \(2008\)](#) were the first to explicitly tackle the prior estimation problem in Positive and Unlabeled Learning as a separate task. They introduce three techniques to estimate c , one of which, henceforth called Elkan’s method (EN), is their recommended choice. The rationale of this method directly derives from [Equation \(3.4\)](#). Precisely, the technique tries to estimate $P(s(\mathbf{x}) = 1 | y(\mathbf{x}) = c_+) = c$ with the two following steps:

In the first step, using both unlabeled U and labeled L datasets together, we train a classification model capable of producing *calibrated* scores, where the class feature is whether the observation belongs to L or not. In other words, the classifier aims to predict $s(\mathbf{x})$ rather than $y(\mathbf{x})$. As the model is a calibrated scorer, it estimates $P(s(\mathbf{x}) = 1 | \mathbf{x})$.

In the second step, in order to estimate $P(s(\mathbf{x}) = 1 | y(\mathbf{x}) = c_+)$ and therefore c , EN uses L as a proxy to the condition of the probability: $y(\mathbf{x}) = c_+$. It averages all scores obtained for the observations in L as follows:

$$\hat{c} = |L|^{-1} \sum_{\mathbf{x}_i \in L} \hat{P}(s(\mathbf{x}_i) = 1 | \mathbf{x}_i)$$

[Figure 40](#) exemplifies Elkan’s algorithm. We make two observations based on these figures. First, positive observations, either labeled or unlabeled, share similar sizes in [Figure 40b](#). Indeed, as they have the same probability distribution, they also share the same

area in the feature space uniformly. In such a case, where features are useless to distinguish labeled observations from positive but unlabeled ones, the best possible estimation for the probability of any single positive observation being labeled is the proportion of labeled observations in the *shared* space, therefore c (see Equation (3.4)).

Figure 40 – Illustration of Elkan’s algorithm. The points on the plot correspond to a uniform sample of 10% of the data used to run the algorithm. The size of the points correlates linearly with the estimated probability of the observation being labeled. Gray dots correspond to unlabeled data (unused to estimate c once the model is trained). Black dots are labeled positive observations. $\hat{c} = 30\%$ is the average of the estimated probabilities of black dots being labeled. In contrast, actual c is 33%.

Source: Elaborated by the author.

The second important aspect to note, in Figure 40b, is that as a negative observation gets farther the positive cluster, it also gets smaller. This happens because they get farther from *labeled* observations, which are the classification target for the model induced. This remark raises the question of what would happen if there were further spatial overlap between the classes. Notice that EN estimates c by averaging $\hat{P}(s(\mathbf{x}_i) = 1 | \mathbf{x}_i)$ for all $\mathbf{x}_i \in L$. This works on the assumption that

$$\begin{aligned} \hat{P}(s(\mathbf{x}_i) = 1 | \mathbf{x}_i) &= \hat{P}(s(\mathbf{x}_i) = 1 | \mathbf{x}_i, y(\mathbf{x}_i) = c_+) \\ &= \hat{P}(s(\mathbf{x}_i) = 1 | y(\mathbf{x}_i) = c_+) \\ &\forall \mathbf{x}_i \in L \end{aligned}$$

While it is true that $y(\mathbf{x}_i) = c_+$ for every observation $\mathbf{x}_i$ in L , we emphasize that the classification model learns how to estimate $\hat{P}(s(\mathbf{x}_i) = 1 | \mathbf{x}_i)$, *not* $\hat{P}(s(\mathbf{x}_i) = 1 | \mathbf{x}_i, y(\mathbf{x}_i) = c_+)$. The true value of the former probability is given according to the following equation:

$$P(s(\mathbf{x}_i) = 1 | \mathbf{x}_i) = P(y(\mathbf{x}_i) = c_+)P(s(\mathbf{x}_i) = 1 | y(\mathbf{x}_i) = c_+)$$

By providing the classifier only with instances from L , EN implicitly assumes that $P(y(\mathbf{x}_i) = c_+) = 1$, whereas it may not be the case. Indeed, $P(s(\mathbf{x}_i) = 1)$ will be significantly lower than $P(s(\mathbf{x}_i) = 1 | y(\mathbf{x}_i) = c_+)$ when there are overlap between the classes, since in such cases $P(y(\mathbf{x}_i) = c_+) \ll 1$. For this reason, when there is overlap between the classes, EN underestimates $\hat{c}$ and therefore overestimates $\hat{p}$.

Newer algorithms handle the possibility of class overlap better than EN by different means.

5.1.1.2 PE and pen-L1

Plessis and Sugiyama (2014) demonstrated that the calculations in EN can be reinterpreted as a minimization of the Pearson divergence (Pd) between $\hat{P}(\mathbf{x})$ and $\alpha\hat{P}(\mathbf{x}|y(\mathbf{x}) = c_+)$, where the former is estimated from U and the latter from L . Finally, they introduced PE, which can be simplified with the expression:

$$\hat{p} = \arg \min_{0 \leq \alpha \leq 1} \text{Pd}(\hat{P}(\mathbf{x}), \alpha\hat{P}(\mathbf{x}|y(\mathbf{x}) = c_+))$$

The major benefit of PE over EN is that the former drops the need for an intermediate model to estimate the posterior probability, whereas the latter needs a calibrated classifier. However, similarly to EN, PE also overestimates the proportion of positive observations whenever there is overlap between the classes. As PE is a reinterpretation of EN and share the same caveat regarding overestimation, we do not detail the method any further.

To circumvent the overestimation of PE, Christoffel, Niu and Sugiyama (2016) introduced pen-L1, which applies a biased and heavy penalization on $\hat{p}$ that implies that in some regions of the feature space $p \times P(\mathbf{x}|y(\mathbf{x}) = c_+) > P(\mathbf{x})$. Such an implication is unrealistic (BEKKER; DAVIS, 2018).

5.1.1.3 AlphaMax

AlphaMax was introduced by Jain *et al.* (2016). In their terminology, U corresponds to the *mixture* sample and L to the *component* sample. The AlphaMax algorithm estimates the maximum proportion of L in U .

To better explain the intuition behind AlphaMax, consider $U_{\oplus} \subseteq U$ contains all positive instances in U , and $U_{\ominus} \subseteq U$ contains all negative instances in U . Finally, consider

$\mathcal{D}(s)$ the density function of the probability distribution of sample s . We know that:

$$\mathcal{D}(U) = (1 - p)\mathcal{D}(U_{\ominus}) + p\mathcal{D}(U_{\oplus}) \quad (5.1)$$

Thanks to the assumption of “selected completely at random”, we also know that $\mathcal{D}(U_{\oplus}) = \mathcal{D}(L)$. In such a case, we can rewrite Equation (5.1) as follows:

$$\mathcal{D}(U) = (1 - p)\mathcal{D}(U_{\ominus}) + q\mathcal{D}(U_{\oplus}) + r\mathcal{D}(L) \quad \forall q, r \in [0, 1], q + r = p \quad (5.2)$$

In Equation (5.2), note that as r increases, q has to proportionally decrease. The objective of AlphaMax is to determine the maximum possible value of r , which is $r = p$ when $q = 0$, for which the equation is still valid. This process is formalized as follows:

$$\hat{p} = \sup_{0 \leq r \leq 1} \{r \mid \mathcal{D}(U) - (1 - p)\mathcal{D}(U_{\ominus}) + q\mathcal{D}(U_{\oplus}) + r\mathcal{D}(L) = 0\}$$

In practice, however, we cannot split U into $U_{\ominus}$ and $U_{\oplus}$, since the data is unlabeled. To circumvent this limitation, AlphaMax constructs two density functions, $\tilde{m}$ and $\tilde{c}$, that re-weight the density functions $\hat{m}$ (which estimates the *mixture* $\mathcal{D}(U)$) and $\hat{c}$ (which estimates the *component* $\mathcal{D}(L)$), according to a shared weight vector ω . We emphasize that $\tilde{m}$ specifically counterbalances $\tilde{c}$ by applying it with $1 - \omega$, similarly to the decomposition of $\mathcal{D}(U)$ in $q\mathcal{D}(U_{\oplus})$. For a given r , AlphaMax proposes an optimization problem to define ω , given the constraint that $\sum \omega_i v_i = r$, where v_i are the weights of $\hat{m}$. For instance, if $\hat{m}$ is estimated using histograms, v_i would be the proportional height of each bin.

The optimization problem tries to maximize a log-likelihood of the mixture (estimation for $\mathcal{D}(U)$) given the weighted participation of the component (estimation for $r\mathcal{D}(L)$). It is stated below:

$$ll_r = \max_{\text{w.r.t. } \omega} \sum_{\mathbf{x} \in U} \log \tilde{m}(\mathbf{x}|\omega) + \sum_{\mathbf{x} \in L} \log \tilde{c}(\mathbf{x}|\omega)$$

Different values of r in the interval $[0, 1]$ are applied in the above optimization problem. While r is lower than p , it is possible for $\tilde{m}$ to counterbalance $\tilde{c}$, keeping the log-likelihood about the same. However, once the applied r is greater than p , the log-likelihood should decrease. AlphaMax returns the value of r that starts the *knee* in the curve of ll_r by r , *i.e.*, the value of r that precedes a steep decrease in ll_r . Figure 41 illustrates that process.

An updated version called AlphaMaxN (JAIN; WHITE; RADIVOJAC, 2016) specifically tackles the possibility of the labeled set containing false-positive observations. This setting is out of the scope of this work. However, we note that in the appendix of

Figure 41 – Illustration of the *knee* finding process in AlphaMax. The red vertical line represents a possible contender for the algorithm to report ($r = 0.65$).

Source: Elaborated by the author.

Jain, White and Radivojac (2016) there is a mathematically detailed description of the AlphaMax algorithm that is more approachable than the description in its original paper.

At last, we emphasize that solving the optimization problem to define ω generally is a computationally intensive task that is required several times (one for each value of r).

5.1.1.4 KM

The algorithms that belong to the KM family (RAMASWAMY; SCOTT; TEWARI, 2016) have a similar rationale to AlphaMax’s. The main difference is that, instead of using log-likelihood to measure the suitability of a possible value for $\hat{p}$ regarding the mixture sample U , they use the distances of kernel embeddings. A better comprehension of the algorithm requires deeper understanding of Reproducing Kernel Hilbert Space, which is out of the scope of this work.

There are two variants of KM: KM1 and KM2. The difference between the variants is the method to select the gradient threshold, which is used in a similar fashion to the “knee” in AlphaMax. KM1 is derived from a theoretical foundation developed by the authors, while KM2 is motivated from empirical evidence.

5.1.1.5 Tree Induction for c Estimation (Tlce)

Tree Induction for c Estimation (Tlce) (BEKKER; DAVIS, 2018), as prior PU algorithms, bases its foundation on the assumption of “selected completely at random”.

Observe that Equation (3.5) can be rewritten as follows:

$$c = \frac{P(s(\mathbf{x}) = 1)}{P(y(\mathbf{x}) = c_+)} \quad (5.3)$$

From Equation (5.3), we can derive that a reasonable estimation $\hat{c}'$ for c is:

$$\hat{c}' = \frac{|L|}{|L| + |U_\oplus|} \quad (5.4)$$

where $U_\oplus \subseteq U$ contains all positive instances in U . However, notice that, as U is unlabeled, we *cannot* directly embed $U_\oplus$ in any equation, in practice.

Nonetheless, from Equation (3.4) we recall that $P(s(\mathbf{x}) = 1)$ is independent of $\mathbf{x}$. In other words, the ratio c in Equation (5.3) is constant for any particular region of the feature space.

Consider $\mathcal{S}_\gamma(A) = \{\mathbf{x} \mid \mathbf{x} \in A, \mathbf{x} \text{ is within region } \gamma\}$ a function that produces a sub-sample of A that contains all observations that are within the region γ of the feature space $\mathcal{X}$. With such a function, we define $\hat{c}_\gamma$ as follows:

$$\hat{c}_\gamma = \frac{|\mathcal{S}_\gamma(L)|}{|\mathcal{S}_\gamma(L)| + |\mathcal{S}_\gamma(U)|} = \frac{|\mathcal{S}_\gamma(L)|}{|\mathcal{S}_\gamma(L)| + |\mathcal{S}_\gamma(U_\oplus)| + |\mathcal{S}_\gamma(U_\ominus)|} \quad (5.5)$$

where $U_\ominus \subseteq U$ contains all negative instances in U .

Finally, TIcE is interested in finding a region γ for which $\hat{c}_\gamma$ approximates $\hat{c}'$, and therefore c . To this end, it needs to downplay the influence of $\mathcal{S}_\gamma(U_\ominus)$. Notice that the region γ that maximizes $\hat{c}_\gamma$ should simultaneously minimize $|\mathcal{S}_\gamma(U_\ominus)|$, since the remaining of the ratio in Equation (5.5) should approximate the constant value c according to the assumption of “selected completely at random”. Therefore, TIcE proposes the following optimization problem:

$$\hat{c} = \max_{\text{w.r.t. } \gamma} \{\hat{c}_\gamma\} - \delta_\gamma$$

where δ_γ is a correction factor (more on that later in this section).

We emphasize that, from the optimization task above, we can derive diverse methods that follow undoubtedly distinct approaches. TIcE, in particular, performs a greedy search by inducing a tree, as we describe next.

In a simplification, to find such a γ , TIcE uses $U \cup L$ to greedily induce a tree where each node is a sub-region of the feature space within the region defined by its parent node. The node that produces the highest $\hat{c}_\gamma$ (given constraints for minimum number of

observations) is used to assess one estimation of c . Several estimations are made via k -fold cross validation and the final one is the average of all estimations assessed.

Tlce is thoroughly described in [Algorithm 3](#), where k is the max-bepp parameter that acts like the Laplace parameter to prefer larger subsets, M is the maximum number of splits and f is the set of folds (obtained with k -fold cross validation). We note that “tree data” and “estimation data” contain both labeled and unlabeled observations.

Algorithm 3 – Tlce

```

1: procedure Tlce( $k, M, f$ )
2:    $\hat{c} \leftarrow 0.5$ 
3:   for  $i \leftarrow 1$  to 2 do
4:      $\hat{c}s \leftarrow \emptyset$ 
5:     for (tree data, estimation data) in  $f$  do
6:        $\delta \leftarrow \max \left\{ 0.025, \frac{1}{1+0.004 \times |\text{estimation data}|} \right\}$ 
7:        $c_{\text{best}} \leftarrow \frac{|\{x : s(x)=1, x \in \text{estimation data}\}|}{|\text{estimation data}|}$ 
8:        $q \leftarrow \{(\text{tree data}, \text{estimation data})\} \triangleright q$  is a priority queue of pairs of sets.
9:        $m \leftarrow M$ 
10:      while  $m > 0$  and  $|q| > 0$  do
11:         $m \leftarrow m - 1$ 
12:         $(S_t, S_e) \leftarrow \arg \max_{(S_t, S_e) \in q} \left\{ \frac{|\{x : s(x)=1, x \in S_t\}|}{|S_t|} - \sqrt{\frac{\hat{c}(1-\hat{c})(1-\delta)}{\delta|S_t|}} \right\}$ 
13:         $q \leftarrow q - \{(S_t, S_e)\}$ 
14:         $a' = \arg \max_{a \in \text{attributes of } S_e} \max_{v \in \text{Dom}(a)} \frac{|\{x : s(x)=1, a(x)=v, x \in S_t\}|}{|\{x : a(x)=v, x \in S_t\}|}$ 
15:        for  $v$  in  $\text{Dom}(a')$  do
16:           $q \leftarrow q \cup \{(\{x : a(x)=v, x \in S_t\}, \{x : a(x)=v, x \in S_e\})\}$ 
17:           $S_a = \{x : a(x)=v, x \in S_e\}$ 
18:           $c_{\text{low}} \leftarrow \frac{|\{x : s(x)=1, x \in S_a\}|}{|S_a|} - \sqrt{\frac{\hat{c}(1-\hat{c})(1-\delta)}{\delta|S_a|}}$ 
19:        end for
20:      end while
21:       $\hat{c}s \leftarrow \hat{c}s \cup \{c_{\text{best}}\}$ 
22:    end for
23:     $\hat{c} \leftarrow \text{average}(\hat{c}s)$ 
24:  end for
25:  return  $\hat{c}$ 
26: end procedure

```

Source: Adapted from [Bekker and Davis \(2018\)](#).

For the purposes of this work, we need to highlight some caveats of the original algorithm.

1. Tlce is introduced as a typical tree-induction algorithm. However, as it uses the estimation assessed by only one node in the tree that may not necessarily be a leaf, it is more accurate to describe it as either a greedy search or a biased optimization

algorithm. Indeed, the algorithm actually intends to locate one region within the feature space.

2. In the original algorithm, a maximum number of splits is set to avoid excessive computing. Regions of the feature-space are processed in order according to a priority-queue. The priority-queue sorts the regions according to how promising they are. Once a region is processed, it is discarded. When TIcE is processing a region, sub-regions are added to the priority-queue. The algorithm only adds sub-regions that are created by dividing the feature-space using only *one* chosen feature. Additionally, the criterion to choose the feature is based solely on the *one* most promising sub-region derived from dividing the space with such a feature alone. It ignores cases where all other sub-regions added are less promising than sub-regions derived from dividing the space with another possible feature. This mechanism implies in severe search bias.
3. We note that δ_γ is always a non-negative number. Indeed, preliminary experiments show that $\hat{c}_\gamma$ tends to overestimate c . However, we believe that the reasoning behind δ_γ is inaccurate. According to Bekker and Davis (2018), the correction relates to the binomial distribution with which each positive observation is labeled (and therefore appears in L): each observation is labeled with a fixed chance c so that the *expectation* of $|L||U_\oplus|^{-1}$ is c , but a difference δ can reasonably occur. However, that difference could go both ways, and the ratio obtained from sample data could be lower than the actual c . In that case, why is δ_γ always non-negative? Further investigation is due. We suspect that the overestimation is actually related to a bias in the way the algorithm selects the sub-regions.
4. The implementation provided by Bekker and Davis (2018) only supports categorical features after binarization. Furthermore, numerical features should be in the range $[0, 1]$. Yet, when a numerical feature is selected to split a node, only four sub-regions are created for the ranges $[0, 0.25]$, $[0.25, 0.5]$, $[0.5, 0.75]$ and $[0.75, 1]$. The feature is not used for subsequent splits.
5. The time complexity of TIcE described by Bekker and Davis (2018) is an overly optimistic $O(mn)$, which is, generally speaking, incorrect. We better estimated the time complexity of TIcE and the full analysis is presented in Appendix A.1. To summarize our findings, the time complexity described by TIcE's original authors is severely underestimated.

In relation to the issue $\#3$, we add that, although L can be seen as a training set, a $|L|$ noticeably larger than $|U|$ can be detrimental. As the algorithm actively tries to maximize the predicted c , a larger $|L|$ turns it easier for the tree-induction algorithm

to overfit: it can more easily find a region that has only labeled data and more than the minimum number of observations required to make a prediction (in which case $\hat{c} = 1$, the maximum possible value). In general, the larger the actual c is, the easier it is for the algorithm to overestimate it. Notably, c grows proportionally to the relation $|L|/|U_{\oplus}|$.

To address issue #2, we suggest an extension for Tlce called Exhaustive Tlce (ExTlce), discussed in [Section 5.2.4](#).

Issue #4 is discussed later, in [Section 5.3](#).

5.1.2 One-class scorers

Our proposals are built on top of distinct rationales. However, they require the use of a base one-class scorer (OCS). A one-class scorer is similar to a binary scorer in the sense that the value (score) issued is correlated with the estimated probability of an observation belonging to the positive class. However, a one-class scorer is trained solely with positive observations, whereas the training set of binary scorers also include negative observations. In this section, we briefly explain the most relevant scorers in literature.

5.1.2.1 Mahalanobis Distance (MD)

The Mahalanobis Distance ([MAHALANOBIS, 1936](#)) is a function that aims to indirectly measure the likelihood of one observation belonging to a sample distribution. It is expressed as a multi-dimensional generalization of counting how many standard deviations a point is afar from the mean of a normal distribution, which applies only to the univariate case.

To calculate the Mahalanobis Distance, first we estimate the distribution of the training sample L as a hyperellipsoid in the feature space X by computing L 's centroid μ and covariance matrix M . For clarity, μ is the average of the observations in L , *i.e.*, a vector whose each dimension is the average of its values in L .

For a given observation $\mathbf{x}$, the distance between $\mathbf{x}$ and the distribution of L is given by:

$$\text{Mahalanobis}(\mathbf{x}, \mu, M) = \sqrt{(\mathbf{x} - \mu) M^{-1} \text{transpose}(\mathbf{x} - \mu)}$$

The Mahalanobis distance is zero when $x = \mu$, and grows as x distances itself from μ along each principal component axis of L . For the sake of completeness, note that the Mahalanobis distance is non-negative, unitless and scale-invariant. We note that to use Mahalanobis distance, one has to assume that the data is clustered within a hyperellipsoid in the original feature space.

Contrary to the definition of score, the Mahalanobis distance is inversely proportional to the probability of x belonging to L . For that reason, in this work we consider the distance multiplied by -1 . Another possible approach would be to consider its inverse.

Figure 42 illustrates a density function of positive scores obtained with the Mahalanobis distance.

Figure 42 – Illustration of a typical density function of one-class scores derived from Mahalanobis distance. The positive class is female *Aedes aegypti*. One negative sub-class (male *Aedes aegypti*) was included only to give a sense of relative scale to the figure. Dataset: Insects v2, described in Section 5.3.6.

Source: Research data.

5.1.2.2 One-class SVM (OSVM)

The basic idea of One-class SVM (OSVM) (KHAN; MADDEN, 2009; NOUMIR; HONEINE; RICHARD, 2012) is to find the smallest hypersphere that contains all training observations within its area. This problem can be formally described as the following optimization task:

$$\min_{\text{w.r.t. } r, \mathbf{c}} r \text{ subject to } \|\mathbf{x}_i - \mathbf{c}\|^2 \leq r^2 \quad \forall \mathbf{x}_i \in L$$

where r is the radius and the vector $\mathbf{c} \in \mathcal{X}$ is the centre of the hypersphere.

Note that at least m observations, where m is the number of dimensions in the feature space $\mathcal{X}$, should lie on the boundary of such a sphere. Those can be seen as analogous to the support-vectors that define the hyperplane in SVM.

In practice, a less restrictive optimization task is used so that the algorithm allows for some of the training observations to lie outside of the sphere to avoid the influence of outliers. Finally, we note that as it is usual for SVM, kernels can be used to make OSVM more flexible and adjustable. Primarily, kernels enable OSVM to better model data that is not concentrated within a hypersphere in the original feature space.

After the hypersphere is induced, to produce scores for previously unseen observations, the model calculates the signed distance of such observations to the boundary of

the hypersphere: positive values are given to observations that lie within the hypersphere, and negative values are issued for observations that lie outside. If the distance is zero, then the observation lies on the boundary. Figure 43 illustrates the shape of the density functions obtained with OSVM as one-class scorer. As we can see in Figure 43b, it can be usual to positive observations that were not in the training set to have negative distances to the hypersphere.

Figure 43 – Illustration of a typical density function of one-class scores derived from One-class SVM with two values for the parameter *gamma* (PEDREGOSA *et al.*, 2011). The positive class is female *Aedes aegypti*. One negative sub-class (male *Aedes aegypti*) was included only to give a sense of relative scale to the figure. Dataset: Insects v2, described in Section 5.3.6.

Source: Research data.

5.1.2.3 Local Outlier Factor

Local Outlier Factor (LOF) (BREUNIG *et al.*, 2000) is typically employed to detect outliers in a single sample of data. However, it can also be used as novelty detector for single observations that do not belong to a training set, and therefore is capable of issuing one-class scores. As this last use is the focus of this thesis, we will explain the algorithm to achieve that end.

The main idea of LOF is to compare the local density of one observation to the local density of its closest neighbors. The score obtained by this observation is proportional to such a relation: if the observation's density is significantly lower than the density of its neighbors, then its score is low, meaning that it has a lower chance of belonging to the training distribution. Otherwise, the score is high, meaning a greater chance of belonging to the training distribution.

Formally, we define $k\text{-distance}_L(\mathbf{x})$ the distance between x and its k -th closest observation in L according to a distance function d (usually Euclidean Distance). Now, consider the reachability-distance as follow:

$$\text{reachability}_{k,L}(\mathbf{x}_i, \mathbf{x}_j) = \max \{k\text{-distance}_L(\mathbf{x}_i), d(\mathbf{x}_i, \mathbf{x}_j)\}$$

Observe that the reachability distance is not a proper distance, since it is asymmetric. The reasoning is that the reachability between $\mathbf{x}_i$ and $\mathbf{x}_j$ is either $k\text{-distance}_L(\mathbf{x}_i)$, if $d(\mathbf{x}_i, \mathbf{x}_j) \leq k\text{-distance}_L(\mathbf{x}_i)$, or $d(\mathbf{x}_i, \mathbf{x}_j)$ otherwise. Note that, for that reason, the reachability distance from $\mathbf{x}$ to any observation within a radius of $k\text{-distance}_L(\mathbf{x})$ from $\mathbf{x}$ is the same. We define $N_{k,L}(\mathbf{x})$ the set of observations within such a hypersphere. Formally:

$$N_{k,L}(\mathbf{x}_i) = \{x_j \mid d(\mathbf{x}_i, \mathbf{x}_j) \leq k\text{-distance}_L(\mathbf{x}_i)\}$$

The *Local Reachability Density* (LRD) is defined as the inverse of the average reachability distance between the neighbors of an observation in the training set L and said observation. Formally:

$$\text{LRD}_{k,L}(\mathbf{x}_i) = \frac{|N_{k,L}(\mathbf{x}_i)|}{\sum_{\mathbf{x}_j \in N_{k,L}(\mathbf{x}_i)} \text{reachability}_{k,L}(\mathbf{x}_j, \mathbf{x}_i)}$$

In the equation above, we emphasize the order of the parameters of $\text{reachability}_{k,L}(\mathbf{x}_j, \mathbf{x}_i)$, since it is not symmetric. Indeed, if the parameters were swapped, the LRD would be the same as $k\text{-distance}_{k,L}(\mathbf{x}_i)$ by definition.

Finally, the Local Outline Factor is calculated as follows:

$$\text{LOF}_{k,L}(\mathbf{x}_i) = \frac{\sum_{\mathbf{x}_j \in N_{k,L}(\mathbf{x}_i)} \text{LRD}_{k,L}(\mathbf{x}_j)}{\text{LRD}(\mathbf{x}_i) |N_{k,L}(\mathbf{x}_i)|}$$

Similarly to the Mahalanobis distance, the value of $\text{LOF}_{k,L}(\mathbf{x})$ is inversely proportional to the probability of belonging to the training data. Therefore, we need to either inverse it or multiply it by minus one in order to convert it to a score. Figure 44 illustrates the behavior of such a score function.

Figure 44 – Illustration of a typical density function of one-class scores derived from Local Outlier Factor. The positive class is female *Aedes aegypti*. One negative sub-class (male *Aedes aegypti*) was included only to give a sense of relative scale to the figure. Dataset: Insects v2, described in Section 5.3.6.

Source: Research data.

One advantage of LOF is that it does not restrict the shape of the data in the feature space. Additionally, as it uses local density measures, it can model training data that has different densities in different regions of the feature space.

We note that, contrary to all other one-class scorers in this section, LOF is *lazy*, that is, it has to keep all training observations in order to issue scores for test observations. Therefore, it may be more restrictive in terms of memory requirements.

5.1.2.4 Isolation Forest

Isolation Forest (LIU; TING; ZHOU, 2008) derives from the use of multiple random decision trees to detect anomalies. The algorithm consists in inducing multiple decision trees whose binary splits are decided randomly (both which feature and the respective decision threshold). The path obtained by traversing such a tree according to a given observation until the leaf, where the observation is isolated (hence the name of the algorithm), tend to be shorter if the observation is an anomaly. Therefore, the score of an observation is inversely proportional to the average of the length of the paths on multiple random trees. Figure 45 illustrates the behavior of a score function derived from Isolation Forest.

Figure 45 – Illustration of a typical density function of one-class scores derived from Isolation Forest. The positive class is female *Aedes aegypti*. One negative sub-class (male *Aedes aegypti*) was included only to give a sense of relative scale to the figure. Dataset: Insects v2, described in Section 5.3.6.

Source: Research data.

5.2 Our proposals

In this thesis, we define One-class Quantification (OCQ) as the task of inducing a quantification model with only positive data. The task is formalized in Definition 8 (REIS *et al.*, 2018a).

Definition 8. A **one-class quantifier** is a quantification model induced from a single-class dataset, in which all available labeled examples belong to the same class, say the positive one, $L = \{(\mathbf{x}_1, c_+), \dots, (\mathbf{x}_n, c_+)\}$, and is defined as

$$q^{\oplus} : \mathcal{X}^? \longrightarrow [0, 1]$$

The one-class quantifier outputs a single probability estimate $\hat{p} \in [0, 1]$ of the positive class prevalence. Notice, however, that $q^{\oplus}$ operates over $\mathcal{X}^?$, *i.e.*, a sample with all occurring classes.

Excluding the explicit objective of inducing a model and disregarding training data afterward, OCQ shares the same purposes of Positive and Unlabeled Prior Estimation, which is detailed in the next section.

Similarly to PU PE, in OCQ we expect the probability distribution of *positive data* to be the same in both training and test sets. After all, in OCQ methods, we induce a *model* that estimates the probability distribution of the positive class, so that we expect to identify and measure the same distribution in an arbitrary test data.

5.2.1 Differences between OCQ and PU PE

Having described OCQ and PU PE, we stress that, from a practical perspective, algorithms from both research areas are capable of solving the same set of problems interchangeably. Therefore, direct comparisons between the methods of such areas are due. However, while both methods *can* solve the same set of problems, there is an essential distinction between the problems that they *aim* to solve. PU PE describes the task as containing exactly two samples: there is no particular interest in modelling a single model that can quantify several test samples. Such a description influenced the development of PU PE techniques, and as a result, all of the examined techniques rely on transductive learning on all stages of the quantification process: they do not produce a single model that can be reused, and a costly process must be done for each and all test samples that need be evaluated.

On the other hand, OCQ methods create a model that estimates the distribution of the positive class, and with which it is possible to quantify any given sample at a later time. As we show in this work, this perspective to the problem provided OCQ techniques with a sizable advantage, in terms of time needed to process a large number of test samples, over PU PE techniques.

We also note that in research work about PU PE, the task is often to estimate either c or $P(y(\mathbf{x}) = c_+)$, whereas in OCQ we are interested in estimating p . In a conversion to the PU PE terminology, $\hat{p}$ estimates $p = P(y(\mathbf{x}) = c_+ | s(\mathbf{x}) = 0)$, *i.e.*, we are interested in estimating the proportion of the positive observations considering *only the unlabeled set*. This divergence is reflected on how the experimental results are shown. We highlight that by measuring the error of estimates of either c or $P(y(\mathbf{x}) = c_+)$, the value obtained is

deflated due to the number of labeled observations (which are training data) influencing the measurement. On the other hand, the size of the training data does not influence evaluation measurements based on $\hat{p}$. We argue that, for this reason, evaluation metrics based on $\hat{p}$ should prevail in both areas of research henceforth.

We can convert the estimation $\hat{c}$ to $\hat{p}$ according to the following equation:

$$\hat{p} = \max \left\{ 1, \frac{\hat{c}^{-1}|L| - |L|}{|U|} \right\}$$

where $|L|$ is the number of labelled observations and $|U|$ is the number of unlabeled observations. The $\max$ in the expression limits the result since the predicted $\hat{c}$ can have a corresponding $\hat{p}$ over one, which would be meaningless for quantification. Observe that $\hat{p}$ is inversely proportional to $\hat{c}$.

Finally, we emphasize that the general assumption of both OCQ and PU PE is that the negative class is unpredictable, but should not fully contain the distribution of the positive class. However, algorithms may impose stricter versions of this assumption to successfully achieve quantification. For instance, Elkan’s algorithm requires clear separation between negative and positive observations.

5.2.2 *Passive-Aggressive Threshold*

Our first proposal, Passive-Aggressive Threshold ACC (PAT-ACC or PAT, for short), draws inspiration from Adjusted Classify and Count and Conservative Average Quantifier (FORMAN, 2006). ACC depends on accurate estimates for TPR and FPR. However, in many applications we cannot reliably measure either TPR and FPR. This is particularly true for tasks derivative from One-class Quantification, since the distribution of scores of *negative* observations varies from sample to sample.

To offer a better grasp on the intuition behind PAT, observe that the influence of the negative distribution on ACC stems from the fact that the most suitable threshold for *classification* usually cut through the density function of the negative scores, leaving negative scores on both sides of the threshold, as seen in Figure 23. Although the number of negative observations on the right-hand side of the threshold is expected to be significantly smaller than on the left-hand side, it is still unpredictable whenever the distribution of negative scores changes.

In PAT, we deliberately choose a very conservative classification threshold that tries to minimize the FPR. In other words, we select a threshold for which we expect very few negative observations to be placed on its right-hand side, as illustrated in Figure 46. With such a conservative threshold, we naively assume that there is no false positive observations. Finally, we extrapolate the total number of expected false negative observations

from the number of true positive observations.

Figure 46 – Expected behavior for a conservative threshold: negligible false positive rate and relative small number of true positive observations, in comparison to the total number of positive observations.

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

More formally, we set the threshold according to a quantile q for the one-class scores of positive observations in a training set. For example, if $q = 0.75$, then the threshold is set so that 75% of the training (positive) observations are scored below such a threshold, while 25% are scored above it. Given q , we estimate $\widehat{TPR} = 1 - q$ and assume $\widehat{FPR} \approx 0$.

After the threshold is set, we perform ACC as usual: we classify all observations in the test sample of size $|U|$ according to this conservative threshold, count the number of positive instances n_+ , estimate the positive proportion $\hat{p}(\oplus) = \frac{n_+}{|U|}$, and readjust it as follows:

$$\hat{p}'(\oplus) = PAT(\hat{p}(\oplus), q) = ACC(\hat{p}(\oplus), 1 - q, 0) = \min \left\{ 1, \frac{\hat{p}(\oplus)}{1 - q} \right\}$$

In PAT, q is an important parameter. Ideally, it should be set as high as possible so that we can be more confident about the assumption of $FPR \approx 0$, even for non-stationary negative distributions. How high it can be set depends on the test sample size, since higher q implies more intense extrapolation from fewer observations. In previous work ([REIS et al., 2018a](#)), we showed PAT's performance to be similar to CC when q approaches 0, as the extrapolation is reduced. We also show that, although important, q is not a sensitive parameter: a broad range of possible values lead to similar quantification errors.

Our previous work showed stability of results for varying q ([REIS et al., 2018a](#)), as illustrated in [Figure 47](#). For that reason, instead of picking a single value q to be used in our experiments, we adopted a strategy similar to Median Sweep ([FORMAN, 2006](#)). In this case, we apply PAT with q from 0.25 to 0.75 with increases of 0.01 and consider the median of the estimates.

Figure 47 – Mean Absolute Error (MAE) of PAT for different values of q in two datasets. The shaded area correspond to two standard deviations.

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

Regarding the time complexity of PAT, since we can reuse the scorer model multiple times, we split the analysis into two stages: training and test.

For the training stage, consider $\mathcal{V}(n)$ to be the time complexity of training a scorer with n observations, and $\mathbf{v}$ the time complexity of scoring one single observation. Suppose that we apply k -fold cross validation to obtain the positive scores, with which we model the density function to identify the t thresholds associated with different values of q . In this case, the complexity to train PAT is the complexity to obtain the scores and sort them in order to identify the thresholds (with binary search):

$$O\left(k\mathcal{V}\left(\frac{(k-1)}{k}|L|\right) + |L|\mathbf{v} + |L|\log|L| + t\log|L|\right)$$

For test, we can take different approaches if we are using multiple thresholds or only one. If we are using only one threshold, then, after scoring all test observations, we can linearly count how many are below the threshold, totalling a time complexity of $O(|U|\mathbf{v} + |U|) = O(|U|\mathbf{v})$. However, if we are using multiple thresholds, we can sort the scores and iterate over a pre-sorted list of thresholds to count how many observations are below each threshold with binary search. In this case, the time complexity is $O(|U|\mathbf{v} + |U|\log|U|)$.

5.2.3 One Distribution Inside

Our second proposal, One Distribution Inside (ODIn), is a Mixture Model (MM) that shares a similar idea with AlphaMax. The main difference between the two algorithms is that ODIn works with univariate distributions (one-class scores), whereas AlphaMax works with multivariate distributions (allowing it to directly work on the feature space).

ODIn searches the maximum possible scale factor $\mathfrak{s}$, $0 \leq \mathfrak{s} \leq 1$, for the known distribution of scores from *positive* training observations, so that it fits inside the distribution of scores from test observations with an overflow no greater than a specified limit. The

overflow is the area between the scaled positive distribution curve and the test distribution curve, where the former is higher than the latter, as illustrated in Figure 48.

Figure 48 – Rationale behind ODIn. The dotted curves represent candidate scale factors for the positive distribution, and the red-shaded area is the overflow area for the greater scale factor (top dotted curve).

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

We represent the distributions as normalized histograms with unit area and b bins, split by $b - 1$ ordered thresholds. The first and last bins are open-ended. This means that all scores lower than the first division fall into the first bin, and all scores higher than the last division fall into the last bin. In our experiments, we set the thresholds between bins, *i.e.*, the values of score that separates the bins, as percentiles obtained from the **positive training observations**. The first and last thresholds are set as the estimates for, respectively, the 0th and 100th percentiles of the scores. The remaining thresholds are set at every $i \times b$ percentile, $0 < i \times b < 100$, $i \in \mathbb{N}$. For instance, if $b = 10$, the thresholds are at the percentiles 0, 10, 20, ..., 90, 100. Although, score wise, the bins do not share the same width, they are expected to be equally filled by observations from the positive distribution. Exceptions are the first and last bins, which are expected to have value close to zero. Figure 49 illustrates this process.

Figure 49 – Thresholds for the histogram bins are not uniformly distributed across the scores (Figure 49a), and yet each bin is filled with the same proportion of data points (Figure 49b).

(a) Distribution of training scores.

(b) Histogram of training scores.

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

The overflow **OF** generated by a histogram H^I , at a scale factor α , inside a histogram H^O , where both histogram are normalized so that $\sum_{1 \leq i \leq b} H_i^I = \sum_{1 \leq i \leq b} H_i^O = 1$, is formally defined as follows:

$$\text{OF}(\alpha, H^I, H^O) = \sum_{i=1}^b \max \left\{ 0, \alpha H_i^I - H_i^O \right\}$$

Given an overflow limit $\mathcal{L}$, which is a parameter, the histogram H^+ with scores for positive training observations, and a histogram H^U with scores for the unlabeled test sample U , ODIn estimates the proportion of positive observations $\hat{p}(\oplus)$ in U as:

$$\hat{p}(\oplus) = \mathfrak{s} - \text{OF}(\mathfrak{s}, H^+, H^U)$$

where

$$\mathfrak{s} = \sup_{0 \leq \alpha \leq 1} \left\{ \alpha \mid \text{OF}(\alpha, H^+, H^U) \leq \alpha \mathcal{L} \right\}$$

Choosing b , although a non-trivial task, is not devoid of useful insights. Histograms with too many bins are negatively affected by two aspects. First, if the sample size is not large enough, histograms with too many bins can become too sparse, each bin can have too low value, and ultimately, the **OF** can face the curse of dimensionality. Second, a large number of bins has the implicit assumption of high precision for the scores. On the other hand, if the number of bins is too small, we may be unable to differentiate distributions. We point the interested reader to the work of [Maletzke et al. \(2019\)](#) for a more in-depth discussion on the effects of the number of bins in a histogram for quantification.

Although $\mathcal{L}$ is a parameter, it can be automatically defined using only positive observations. To this end, we estimate the mean $\hat{\mu}$ and standard deviation $\hat{\sigma}$ of **OF** for pairs of histograms derived from samples with only positive observations, at scale factor 1, and set $\mathcal{L} = \hat{\mu} + d\hat{\sigma}$, where d is a parameter. Although we are actively replacing one parameter with another one, d has a clearer semantic and its value is domain independent: it is the number of standard deviations of the expected average overflow.

Similarly to PAT, the time complexity of ODIn should be split into training and test stages. For training, one has to produce the scores using k -fold cross validation and create the histogram. This implies sorting the scores to find out the percentiles that split the bins. Therefore, the time complexity of training is:

$$O \left(k^{\mathcal{V}} \left(\frac{(k-1)}{k} |L| \right) + |L| \mathfrak{v} + |L| \log |L| \right)$$

For the test stage, one has to score all unlabeled observations and fill in the histogram, accordingly. Finally, we note that $\mathfrak{s}$ can be found through Binary Search, with a time complexity of $O\left(\frac{100}{h} \log_2 \varepsilon^{-1}\right)$, where ε is the expected precision. Therefore, the time complexity of the test stage is:

$$O\left(|U|v + \frac{100}{h} \log_2 \varepsilon^{-1}\right)$$

5.2.4 Exhaustive TIcE

In this section, we introduce a minor modification of TIcE, called Exhaustive TIcE (ExTIcE), that intends to decrease TIcE's search bias. We note that with ExTIcE we only intend to reduce the bias caused by the greedy search. The bias towards regions with higher c is nonetheless maintained.

The main distinction from TIcE is that ExTIcE adds all sub-regions created by splitting the feature space with all features into the priority-queue. ExTIcE is considerably slower for that reason. If the quantification errors obtained by ExTIcE are significantly smaller than those obtained by TIcE, we can infer that there is room for improvements in the way TIcE searches the feature-space, which can be too biased for now.

In ExTIcE, we modify TIcE by replacing lines 14 to 19 of Algorithm 3 with the contents of Algorithm 4.

Algorithm 4 – ExTIcE

```

1: for  $a'$  in attributes of  $S_e$  do
2:   for  $v$  in  $\text{Dom}(a')$  do
3:      $q \leftarrow q \cup \{(\{x : a(x) = v, x \in S_t\}, \{x : a(x) = v, x \in S_e\})\}$ 
4:      $S_a = \{x : a(x) = v, x \in S_e\}$ 
5:      $c_{\text{low}} \leftarrow \frac{|\{x : s(x)=1, x \in S_a\}|}{|S_a|} - \sqrt{\frac{\hat{c}(1-\hat{c})(1-\delta)}{\delta|S_a|}}$ 
6:   end for
7: end for

```

5.3 Experimental setup

In this section, we explain the experimental setup and datasets used in our empirical evaluation. In the general setting, for each dataset, we varied the true positive ratio, *i.e.*, the proportion of the positive class in the unlabeled (test) sample, from 0% to 100% with increments of 10%. For a given positive ratio, we performed 5-fold cross validation to generate candidate observations for training and for test. The (labeled) training and (unlabeled) test samples are drawn, without replacement, from the training and test candidate sets, respectively. The training sample only includes positive observations, and the test sample obeys the positive ratio previously set.

We note that, as is the case with the experiments of [Bekker and Davis \(2018\)](#), and contrary to typical experimental settings, the smaller fold of the data (one fifth) is used for training, while the larger (four fifths) is used for testing. In that case, one single test observation may appear across multiple test samples. However, it will not appear more than once in a sample. This usage of the 5-fold cross validation is employed due to the amount of data required to create test samples with varying proportions of positive observations and negative sub-classes. Due to the slowness of some of the algorithms tested, training data size was limited to 500 observations and test data size was limited to 2,000 observations.

The whole process was repeated one, five or 30 times, depending on the experiment, so that we generate varied samples. Final results are reported as Mean Absolute Error (MAE), which is the average of the absolute difference between the predicted positive ratio and the true positive ratio.

Finally, we raise attention to the fact that if we employ a quantifier that always predict $\hat{p} = 0.5$, and the actual p is uniformly distributed within the range $[0, 1]$ (as in our experiments), then the MAE obtained in a large enough number of test samples converges to 0.25. This fact indicates that the maximum error we should consider as acceptable in our setting is 25%.

All code is available in our supplemental material website ([REIS et al., 2019](#)). Next, we describe the particularities of each experiment.

5.3.1 Experiment #1

In this experiment, the existence of negative sub-classes is disregarded. The size of the test sample is the minimum between the number of available positive and negative candidates, limited to 2,000. The number of repetitions is five. This experiment is designed to be easy to reproduce and compare, although supports only a superficial analysis of performance. Our objective with this experiment is to provide a similar setup with those in current literature, and to provide a general analysis of quantification performance. We increment our analysis with other more specific experiments.

5.3.2 Experiment #2

In this experiment, the existence of negative sub-classes is acknowledged. For each test sample, the proportion of the negative sub-classes is randomized and the sample is drawn accordingly. The size of the test sample is the largest that make the previously set proportion viable, limited to a maximum of 500 observations. To obtain greater variability in the test samples, giving the random proportion of sub-classes, the number of repetitions is 30. With this setting, we aim to produce experimental results that better suit our

assumption that the negative class is unpredictable.

5.3.3 Experiment #3

The uniform randomization of the proportions of negative sub-classes, in Experiment #2, has an adverse effect. While the MAE for each individual sub-class proportion is informative for the expected performance for said proportion, the experimental MAE when averaging all variations of sub-class proportions is bound to converge to the same MAE that would be obtained with balanced test samples, that is, test samples whose every single sub-class has the same number of observations.

However, in real world applications, we do not assume that all classes will appear with the same proportion. On the contrary, we assume that the behavior is unpredictable. To better evaluate the methods in this situation, we propose Experiment #3. In this experiment, we map the original dataset onto several datasets, one *for each* negative sub-class, containing data points of said sub-class and all positive data points. Each dataset is evaluated individually. The size of the test samples is the minimum between the number of available positive and negative candidates, limited to 2,000. Finally, we evaluate:

Experiment #3-a – Median half of the negative classes produced MAE lower or equal than the one reported in this experiment;

Experiment #3-b – 75-percentile three quarters of the negative classes produced MAE lower or equal than the one reported in this experiment;

Experiment #3-c – Worst case the result obtained by the single negative class that produced the greatest MAE.

5.3.4 Experiment #4

The aim of this experiment is to compare execution time of different algorithms.

Due to the slowness of some of the algorithms evaluated, the previous experiments were executed in parallel in a variety of hardware across multiple days. To measure the time consumed by each algorithm in a comparable manner, we proposed a diminished version of Experiment #1 that was executed in a single machine. The differences are: 5-fold cross validation was interrupted after the evaluation of the first fold, and the experimental setup was evaluated only once instead of repeating for five times.

We highlight that the time necessary to quantify each test sample was measured independently, and summed at the end, to avoid measuring time spent with the preparation of the samples.

5.3.5 Algorithms

We evaluated the performance of several algorithms: EN, PE, KM1, KM2, TIcE, ExTIcE, and PAT. All methods were merged into a unified test framework, publicly available as supplemental material.

Given the algorithm’s simplicity, we used our own implementation for EN. [Elkan and Noto \(2008\)](#) employed Support Vector Machine calibrated with Platt scaling as a base classifier for EN. In this work, we adopted the same method to keep compatibility between experiments.

EN relies on SVM. We used scikit-learn’s implementation ([PEDREGOSA *et al.*, 2011](#)) with all parameters set to default, excepting gamma. Gamma is an important parameter that usually is set to either “auto” or “scale”. This parameter caused severe differences in the results for some datasets. For this reason, we report results for both settings, where EN_a refers to the situation where gamma is set to “auto”, and EN_s refers to the situation where gamma is set to “scale”.

The code of PE, used in our experiments, was a direct translation to Python 3 from the original code, in Matlab, provided by [Plessis and Sugiyama \(2014\)](#)¹.

Code for pen-L1 is not available in the author’s website. However, comparisons are possible due to transitivity and analysis of previous work ([BEKKER; DAVIS, 2018](#)).

We reached out to the AlphaMax’s authors and they attentively provided us with code and instructions to use AlphaMax in our experiments. Unfortunately, a fair use of the program provided would require several manual interventions. Given the volume of the experiments in our setup, making such interventions would be unfeasible and unfair with the other contenders. Alternatively, results for AlphaMax are provided in previous work ([RAMASWAMY; SCOTT; TEWARI, 2016](#); [BEKKER; DAVIS, 2018](#)), so that it is possible to draw some conclusions by assuming transitivity.

For KM1 and KM2, we used code provided by their original authors ([RAMASWAMY; SCOTT; TEWARI, 2016](#))².

The code of KM1 and KM2 is a single script that produces results for both variants, since they share the significant part of the computation required to evaluate a sample. For this reason, in Experiment #4, time spent for both algorithms is aggregated into a single column, KM.

Although we previously tested PAT with different scorers ([REIS *et al.*, 2018a](#)), in our analysis, we keep only the results for PAT with Mahalanobis distance (PAT_M). We chose PAT_M to be a representative of PAT in our comparisons against PU techniques since

¹ [<http://www.mcduplessis.com/index.php/>](http://www.mcduplessis.com/index.php/)

² [<http://web.eecs.umich.edu/~cscott/code.html>](http://web.eecs.umich.edu/~cscott/code.html)

Mahalanobis Distance is the simplest scorer among the ones cited in this work and does not require any parameter, and having one single version of PAT simplifies our analysis. Another important difference regarding our previous usage of PAT is that, here, we vary the parameter q from 25% to 75% with increments of one and report the median of all predictions, instead of fixing the parameter to a single value.

As PAT_M is the only algorithm tested that produces a model that can be used for several test samples, in Experiment #4 we *additionally* report the time spent by PAT_M to only quantify the data, while disregarding the time spent with training.

PAT and ODIn were preliminarily compared in the same setting proposed in Experiment #1, with both methods adopting Mahalanobis distance. PAT was consistently superior to ODIn. As both methods are based on the same rationale of learning the distribution of one-class scores, we kept only PAT in our evaluation against the PU PE techniques, considering it as a representative of such a general approach. The comparison between PAT and ODIn can be found in [Appendix A.2](#).

In our experiments, we only used numerical datasets. Hence, to address TlEc’s issue #4, which states how the original authors’ implementation handle numerical data too simplistically, we developed our own implementation. Our implementation is meant to be used only with numerical features. For each split, we divide the region into two sub-regions with roughly the same number of observations: one with all observations that are below or equal the median of the splitting feature, and the other with the remaining. The chosen feature can still be used subsequently. We note that we sort the data to compute the median for each attribute that is evaluated as a split candidate, and we allow for a feature to be used more than once. The sorting could be avoided by keeping simultaneous presorted arrays with references to the observations.

Throughout experiments #1 to #3, we additionally compare all algorithms against a hypothetical classify and count approach that uses the Mahalanobis distance as scorer and the best possible threshold for classification. We call such an algorithm Best Fixed Threshold with Mahalanobis (BFT_M). To choose the threshold, we evaluate several thresholds based on the percentiles of the positive training data (from 0 to 100, with increments of 1). For each dataset, we evaluate which threshold generated the lowest MAE on the test samples and report such a result. We emphasize that, regardless of the average performance obtained by BFT_M , it still is affected by the systemic error explained in [Section 5.2](#).

5.3.6 Datasets

In our experiments, we used 10 datasets. Nine are directly derived from real data, and one is generated by a Bayesian network. To maintain consistency, for each dataset, the positive class is the same as in our previous work ([REIS et al., 2018a](#)), where they

were chosen arbitrarily. Each dataset is detailed below:

Insects v2 sensor data regarding the flight of 18 classes of insects. A class of insect is determined by sex and species. The observations are described by 10 features extracted from a time series obtained from a single sensor. No environmental feature is included. All data was collected within a temperature range from 27 (included) to 30 (included) degree Celsius. The number of observations per class was limited to 10,000 observations (achieved by seven classes). The class with the least number of observations has 259. The total number of records is 83,550, and the positive class is female *Aedes aegypti* with 10,000 observations;

Insects contains information about the flight of 14 species of insects. As some are discriminated further by sex, the dataset has 18 classes. The positive class is female *Aedes aegypti*. The data has 166,880 records represented by 27 features. We find this dataset to be heavily biased regarding the environmental feature temperature, while the features that should describe the flights themselves are not very informative. This dataset was kept in our evaluations only to maintain consistency with our previous work (REIS *et al.*, 2018a), but is now considered to be deprecated for practical purposes;

Arabic Digit contains 8,800 entries described by 26 features for the human speech of Arabic digits. There are 10 classes, and the target class is the digit 0. This version sets a fixed number of features for every record (HAMMAMI; BEDDA, 2010; LICHMAN, 2013);

BNG (Japanese Vowels) Bayesian network generated benchmark dataset with speech data regarding Japanese Vowels. There are 1,000,000 entries, represented by 12 features, for 9 speakers. The speaker #1 is the class of interest (VANSCHOREN *et al.*, 2013);

Anuran Calls (MFCCs) contains 22 features to represent the sound produced by different species of Anurans (frogs). As the data size is restricted, we only considered the two biggest families of frogs as the classes of the data, ending up with 6,585 entries. The positive class is the *Hylidae* family, and the negative class is the *Lepidodactylidae* family (DIAZ *et al.*, 2012; LICHMAN, 2013);

Handwritten contains 63 features that represent the handwritten lowercase letters *q*, *p* and *g*. The data has 6,014 entries and the chosen positive class is the letter *q* (REIS; MALETZKE; BATISTA, 2018);

Letter describes the appearance of the 26 uppercase letters of the alphabet on a black and white display with 16 features. It contains 20,000 entries and the class of interest is the letter *W* (FREY; SLATE, 1991; VANSCHOREN *et al.*, 2013);

Pen-Based Recognition of Handwritten Digits handwritten digits represented by 16 features. The digit 5 is the target class. There are 10,992 entries (ALIMOGLU; ALPAYDIN; DENIZHAN, 1996; LICHMAN, 2013);

HRU2 Pulsar candidates collected during the HTRU survey, where pulsars are a type of star. It contains two classes, Pulsar (positive) and not-Pulsar (negative), across 17,898 entries described by 63 features (LYON *et al.*, 2016; LICHMAN, 2013);

Wine Quality contains 11 features that describe two types of wine (white and red). The quality information was disregarded, and the target class is red wine. The dataset contains 6,497 entries (CORTEZ *et al.*, 2009; LICHMAN, 2013).

We note that KM1 and KM2 presented a runtime error while processing dataset H (Handwritten).

5.4 Experimental evaluation

In this section, we display and analyze the results we obtained with the experiments explained in the previous section. For all experiments, we present the average rank and, from completeness, a critical difference plot for the Nemenyi test with $\alpha = 0.1$. This test is intended as a simple way of comparing all algorithms in one go. However, we observe the limitations of this test as it only takes the ranks into account and is conservative with the amount of data we have. In some cases, the difference between some results are glaring, even in different orders of magnitude, and the test fails to recognize the superiority of some approach. We make particular observations for such cases and perform pair-wise comparisons via Wilcoxon signed-rank test, when relevant.

Table 14 summarizes our results for Experiment #1, and Figure 50 shows the corresponding critical difference plot. PAT_M , our proposal, outperformed all PU approaches in 9 datasets out of 10. It underperformed (within one standard deviation) KM1 and ExTicE only in the dataset Insects v2, in which PAT_M ranked third. We observe that, as expected, PAT_M outperformed BFT_M in most cases. Although BFT_M is overly optimistic since the threshold is chosen based on the final results, it still undergoes CC's systemic error explained in Section 3.2.3.

Also as expected, ExTicE outperformed TicE in every dataset, since ExTicE removes a search constraint from TicE. However, more noteworthy is the fact that ExTicE performed better than all PU Learning approaches in all datasets, although a direct comparison against BFT_M is inconclusive (p-value is one for Wilcoxon Rank-Sum test).

Regarding Figure 50, we note that although PAT_M did not differ significantly from ExTicE, the test only evaluated the average rank of the algorithms. Directly comparing

Table 14 – Mean absolute error (standard deviation in parentheses), in percentages, for experiment #1.

Data	EN _a	EN _s	PE	KM1	KM2	TlCE	ExTlCE	PAT _M	BFT _M
N	37.02 (22.99)	14.60 (6.42)	15.94 (3.94)	7.98 (5.10)	17.06 (4.46)	16.59 (2.73)	7.72 (2.78)	8.92 (2.43)	8.45 (1.49)
I	26.92 (2.81)	25.62 (4.85)	20.55 (1.51)	12.91 (4.76)	14.63 (4.41)	21.72 (3.56)	12.78 (3.55)	7.34 (2.48)	10.28 (1.71)
A	10.78 (5.47)	11.14 (4.88)	13.34 (6.15)	25.43 (10.25)	26.39 (6.92)	20.49 (4.77)	13.12 (3.40)	3.87 (2.79)	6.45 (2.40)
B	13.98 (3.79)	10.94 (2.93)	16.66 (2.35)	13.37 (8.25)	18.48 (5.08)	17.25 (2.76)	8.85 (3.06)	5.76 (2.37)	10.04 (1.81)
C	14.86 (4.82)	12.64 (4.63)	11.79 (2.09)	12.02 (5.71)	15.25 (3.92)	11.32 (2.07)	4.26 (2.30)	2.03 (1.63)	10.85 (3.77)
H	8.03 (3.34)	49.02 (1.90)	12.57 (3.08)	–	–	11.37 (2.04)	5.68 (2.61)	3.66 (4.69)	4.76 (3.32)
L	8.97 (6.95)	10.42 (6.24)	12.49 (5.71)	16.02 (8.22)	19.57 (5.64)	10.23 (3.39)	5.84 (3.19)	3.18 (2.33)	2.99 (1.66)
P	50.04 (3.59)	10.58 (5.56)	10.75 (3.67)	12.86 (6.94)	20.80 (4.27)	13.45 (4.16)	6.81 (3.31)	2.57 (1.98)	2.71 (1.47)
R	24.19 (11.21)	14.46 (4.73)	15.44 (2.94)	6.46 (4.77)	10.47 (3.07)	7.86 (2.13)	4.17 (2.57)	2.64 (2.09)	11.76 (3.42)
W	12.12 (5.31)	21.98 (4.86)	16.20 (2.37)	5.45 (3.91)	8.77 (4.28)	8.56 (1.97)	3.23 (2.46)	2.23 (1.87)	4.94 (2.18)
rank	6.8	6.1	6.4	5.5	7.7	5.9	2.6	1.4	2.7

Source: Research data.

Figure 50 – Nemenyi test for Experiment #1 with $\alpha = 0.1$. Methods within the critical difference (CD) are connected by horizontal line and are not significantly different.

Source: Research data.

PAT_M against ExTlCE with Wilcoxon Rank-Sum test results in a p-value of 0.2.

The results from Experiment #2 were unremarkably similar to the ones from Experiment #1. This is due to the fact that the majority of the datasets used in our experiments are already fairly balanced (regarding the negative sub-classes). For this reason, we do not further analyze such results. They are displayed in [Appendix A.3](#).

[Table 15](#) and [Figure 51](#) present the results for Experiment #3-a. As a recap, the results in the table indicates that, for half the classes, the MAE obtained is lower or equal than the value shown. While the rankings are mostly unchanged from Experiments #1 and #2, we observe that for some datasets, especially N, I, and B, the MAE committed by

both ExTlCE and PAT_M are below half of those committed in the previous experiments. This is evidence of a great disparity in the separability between different sub-classes and the positive class. We can therefore expect that the PAT's low errors in Experiment #3-a should be compensated by bigger errors as we investigate larger percentiles, in Experiments #3-b and #3-c.

Table 15 – Mean absolute error (standard deviation in parentheses), in percentages, for experiment #3-a. Results for datasets C, R and W are repetitions from Table 14, since they contain only one negative sub-class.

Data	EN _a	EN _s	PE	KM1	KM2	TlCE	ExTlCE	PAT_M	BFT _M
N	32.82 (12.49)	8.50 (5.54)	15.98 (1.89)	11.18 (5.19)	14.48 (2.78)	7.93 (1.27)	3.53 (1.81)	2.09 (1.75)	5.29 (1.70)
I	20.05 (3.02)	20.36 (2.97)	16.17 (1.32)	7.45 (3.54)	9.12 (2.81)	10.92 (1.42)	5.23 (2.35)	2.16 (1.67)	7.50 (2.04)
A	8.54 (5.80)	9.65 (6.12)	13.69 (5.09)	14.30 (7.72)	19.33 (5.38)	10.81 (3.59)	6.44 (2.28)	3.73 (2.69)	6.79 (2.53)
B	10.26 (3.18)	7.72 (2.93)	13.51 (1.79)	10.23 (4.53)	14.81 (4.42)	11.22 (2.12)	4.89 (2.17)	3.21 (1.99)	9.07 (2.01)
C	14.86 (4.82)	12.64 (4.63)	11.79 (2.09)	12.02 (5.71)	15.25 (3.92)	11.32 (2.07)	4.26 (2.30)	2.03 (1.63)	10.74 (3.63)
H	7.93 (3.28)	49.57 (1.05)	11.87 (2.46)	–	–	11.12 (1.97)	5.49 (2.34)	3.34 (3.76)	6.48 (3.76)
L	8.03 (5.38)	10.26 (5.69)	11.93 (4.42)	12.09 (6.81)	18.20 (5.31)	6.53 (2.86)	3.73 (2.53)	3.07 (2.38)	2.55 (1.17)
P	49.84 (1.01)	11.92 (5.63)	10.09 (3.39)	12.29 (5.18)	17.04 (3.80)	7.84 (3.18)	4.62 (2.68)	2.58 (2.11)	2.54 (0.94)
R	24.19 (11.21)	14.46 (4.73)	15.44 (2.94)	6.46 (4.77)	10.47 (3.07)	7.86 (2.13)	4.17 (2.57)	2.64 (2.09)	11.49 (3.29)
W	12.12 (5.31)	21.98 (4.86)	16.20 (2.37)	5.45 (3.91)	8.77 (4.28)	8.56 (1.97)	3.23 (2.46)	2.23 (1.87)	4.95 (2.21)
rank	6.9	6.4	6.9	5.8	7.5	4.9	2.2	1.2	3.1

Source: Research data.

Figure 51 – Nemenyi test for Experiment #3-a with $\alpha = 0.1$. Methods within the critical difference (CD) are connected by horizontal line and are not significantly different.

Source: Research data.

Table 16 and Figure 52a present the results for Experiment #3-b, and Table 17 and Figure 52b, for Experiment #3-c. Whereas for 75-percentile (Experiment #3-b) PAT_M still maintains significantly lower MAE than ExTlCE in pairwise comparison (p-value of

Table 16 – Mean absolute error (standard deviation in parentheses), in percentages, for experiment #3-b. Results for datasets C, R and W are repetitions from Table 14, since they contain only one negative sub-class.

Data	EN _a	EN _s	PE	KM1	KM2	TlCE	ExTlCE	PAT _M	BFT _M
N	40.02 (21.10)	10.05 (5.96)	27.04 (1.43)	12.25 (4.14)	18.01 (4.08)	8.62 (1.33)	5.05 (2.76)	3.17 (2.58)	7.78 (2.84)
I	21.40 (2.69)	26.03 (3.01)	20.37 (1.50)	10.68 (4.14)	12.36 (4.49)	11.58 (2.12)	5.68 (2.43)	6.99 (3.03)	15.30 (3.00)
A	10.02 (5.34)	10.60 (5.66)	14.64 (4.02)	15.96 (8.28)	22.23 (5.14)	12.13 (3.82)	7.32 (2.50)	4.25 (3.13)	8.22 (3.23)
B	15.78 (3.93)	12.28 (3.67)	15.73 (2.41)	12.40 (7.05)	16.61 (4.68)	14.04 (2.62)	7.14 (2.78)	16.46 (3.99)	20.59 (2.28)
C	14.86 (4.82)	12.64 (4.63)	11.79 (2.09)	12.02 (5.71)	15.25 (3.92)	11.32 (2.07)	4.26 (2.30)	2.03 (1.63)	10.74 (3.63)
H	7.93 (3.28)	49.57 (1.05)	11.87 (2.46)	–	–	11.12 (1.97)	5.49 (2.34)	3.34 (3.76)	6.48 (3.76)
L	8.32 (5.72)	10.88 (5.79)	12.41 (4.58)	13.28 (9.12)	18.96 (5.85)	7.42 (3.34)	4.11 (2.54)	3.19 (2.75)	2.85 (1.40)
P	50.02 (8.59)	12.20 (5.76)	10.74 (3.88)	13.46 (5.32)	18.01 (4.34)	8.47 (2.81)	4.98 (2.48)	2.65 (2.08)	2.56 (0.96)
R	24.19 (11.21)	14.46 (4.73)	15.44 (2.94)	6.46 (4.77)	10.47 (3.07)	7.86 (2.13)	4.17 (2.57)	2.64 (2.09)	11.49 (3.29)
W	12.12 (5.31)	21.98 (4.86)	16.20 (2.37)	5.45 (3.91)	8.77 (4.28)	8.56 (1.97)	3.23 (2.46)	2.23 (1.87)	4.95 (2.21)
rank	6.9	6.3	6.6	5.7	7.5	4.4	2.0	1.9	3.8

Source: Research data.

Figure 52 – Nemenyi test for Experiments #3-b and #3-c with $\alpha = 0.1$. Methods within the critical difference (CD) are connected by horizontal line and are not significantly different.

(a) Experiments #3-b.

(b) Experiments #3-c.

Source: Elaborated by the author.

0.03 according to Wilcoxon Rank-Sum test), the opposite takes place for 100-percentile (Experiment #3-c). In fact, due to the poor performance of PAT_M in datasets N, I, and B, its average rank was 2.7 despite the fact that the algorithm ranked first in all other datasets. Finally, although the performance of ExTlCE for the same datasets decreased in comparison to the previous experiments, it still outperformed all other approaches. In the remaining datasets, ExTlCE ranked second, only behind PAT_M.

Particularly for dataset N, observe that the average error obtained by PAT_M is close to 50% in Table 17. As the actual positive ratio varied uniformly within the interval

Table 17 – Mean absolute error (standard deviation in parentheses), in percentages, for experiment #3-c. Results for datasets C, R and W are repetitions from Table 14, since they contain only one negative sub-class.

Data	EN _a	EN _s	PE	KM1	KM2	TlCE	ExTlCE	PAT _M	BFT _M
N	49.87 (10.17)	18.89 (6.36)	38.49 (1.07)	18.07 (4.09)	22.24 (4.99)	17.18 (3.17)	7.69 (3.24)	49.95 (0.76)	44.11 (1.09)
I	28.14 (3.27)	38.08 (3.97)	24.81 (2.41)	16.41 (6.12)	18.50 (6.22)	22.16 (3.02)	12.67 (3.22)	38.28 (4.42)	33.04 (3.24)
A	10.83 (5.65)	11.23 (6.57)	15.18 (6.71)	19.91 (9.95)	25.78 (5.35)	15.03 (4.01)	9.79 (2.95)	5.68 (3.67)	12.92 (3.73)
B	19.99 (3.92)	17.19 (3.91)	18.57 (2.25)	18.90 (8.21)	21.33 (5.93)	17.48 (2.76)	9.10 (2.81)	16.59 (3.57)	21.98 (2.87)
C	14.86 (4.82)	12.64 (4.63)	11.79 (2.09)	12.02 (5.71)	15.25 (3.92)	11.32 (2.07)	4.26 (2.30)	2.03 (1.63)	10.74 (3.63)
H	7.93 (3.28)	49.57 (1.05)	11.87 (2.46)	–	–	11.12 (1.97)	5.49 (2.34)	3.34 (3.76)	6.48 (3.76)
L	8.81 (6.07)	12.95 (5.69)	13.11 (5.09)	16.86 (9.60)	23.82 (5.79)	9.10 (3.59)	5.83 (3.16)	3.65 (2.47)	19.83 (6.07)
P	50.40 (1.81)	13.71 (5.99)	11.14 (3.66)	14.69 (6.29)	19.43 (5.83)	11.74 (3.75)	6.65 (3.02)	2.85 (2.25)	13.29 (6.74)
R	24.19 (11.21)	14.46 (4.73)	15.44 (2.94)	6.46 (4.77)	10.47 (3.07)	7.86 (2.13)	4.17 (2.57)	2.64 (2.09)	11.49 (3.29)
W	12.12 (5.31)	21.98 (4.86)	16.20 (2.37)	5.45 (3.91)	8.77 (4.28)	8.56 (1.97)	3.23 (2.46)	2.23 (1.87)	4.95 (2.21)
rank	6.4	6.0	5.9	5.5	7.0	4.2	1.7	2.7	5.6

Source: Research data.

$[0, 1]$ during the experiment, such an error indicates that PAT_M always predicted $\hat{p}$ as either close to zero or close to one. Considering our previous results for PAT_M in this same dataset, we can infer that the current situation corresponds to the latter case, since the algorithm could previously detect situations where the positive class was not prominent (the error was below the baseline 25%), and the learning process involved only the positive class. In fact, further analysis of our more detailed data (available as supplemental material) reveals that the average prediction of p was 99.74%, which indicates that observations from the negative class obtained score values at least as large as those of positive observations. From this piece of data, we can assume that observations that belong to this negative class are highly similar to at least part of the positive data, fact that also affected the best classify and count BFT_M. In the next section, we discuss how and why this scenario affected PAT_M to a considerably greater degree than ExTlCE. Before that, we do our final analysis regarding time consumption.

Table 18 presents the total time, in seconds, required to perform all tasks in Experiment #4. We can see that PE was several orders of magnitude slower than the other approaches. KM and predictably ExTlCE were both orders of magnitude slower than TlCE, PAT_M and EN. Although TlCE, EN and PAT_M were generally in the same order of magnitude, PAT_M performed consistently faster, even when the time necessary to train the scorer is considered.

Given the proposed experimental setup, we cannot conclusively claim that EN, TlCE and PAT always have numerically similar execution times. We note that the training

Table 18 – Total time spent, in seconds, to accomplish all tasks in experiment #4. As PAT_M is the only algorithm that produces a model that can be reused for several test samples, one additional column shows the time spent by PAT_M disregarding the training stage.

Data	EN _a	EN _s	PE	KM	TlCE	ExTlCE	PAT_M	PAT_M w/o T
N	20.19	8.41	2960.44	227.17	4.50	84.59	2.55	1.82
I	15.80	6.67	2971.05	240.50	13.68	232.96	2.58	1.84
A	1.30	1.27	212.41	23.12	4.78	142.62	1.03	0.67
B	6.53	9.89	2959.61	209.72	6.15	110.52	2.61	1.79
C	5.30	6.44	1432.50	213.06	7.89	139.28	2.16	1.65
H	9.86	5.56	1231.82	–	23.78	339.50	2.17	1.47
L	1.08	0.78	253.69	15.10	3.72	95.65	1.05	0.68
P	4.07	1.40	518.00	41.02	4.48	114.10	1.37	0.94
R	8.26	2.53	773.14	150.89	2.35	45.23	1.68	1.16
W	6.01	2.07	745.12	125.60	3.17	59.29	1.58	1.22
rank	4.3	3.4	8.0	6.6	4.2	6.4	2.1	1.0

Source: Research data.

dataset was limited to 500 observations, and the test sample to 2,000. We believe that further experimentation would have shown both EN and TlCE to become several orders of magnitude slower than PAT_M for bigger samples due to the time complexities of SVM and TlCE. Additionally, replacing PAT_M 's Mahalanobis Distance with a different dissimilarity function would also impact its execution time performance.

5.5 Related additional contributions

In this section, we list and detail our additional contributions that are related to One-class Quantification.

5.5.1 One-class Naive Bayes

In this section, we introduce the One-class Naive Bayes (ONB), which is a one-class scorer directly inspired by the simplest version of Naive Bayes for nominal data, despite ONB being designed for numerical data.

For each attribute of the data, *i.e.*, each dimension of the feature space $\mathcal{X}$, we build a histogram with b bins using only the training data. All bins in the histogram of one attribute are of same size, *i.e.*, cover a same interval length. The value of one bin is the proportion of how many observations have the corresponding attribute within its covered interval. This means that the histogram is normalized so that the sum of the bins is one. Therefore, each bin corresponds to the estimate of the probability of one observation's attribute value lying in the interval covered by such a bin. Finally, the score of an unseen observation $\mathbf{x}$ is the estimate probability of sampling an observation whose all attributes are within the same intervals as $\mathbf{x}$. Assuming that all attributes are conditionally independent, the score is calculated as follows:

$$\begin{aligned}
ONB(H, \mathbf{x}) &= \prod_{i=0}^m H_i(attr_i(\mathbf{x})) \\
&= \exp \sum_{i=0}^m \log(H_i(attr_i(\mathbf{x})))
\end{aligned}$$

We emphasize the difference between One-class Naive Bayes and Gaussian Naive Bayes: the former discretizes each feature into ranges, and therefore computes probabilities rather than likelihoods. The latter computes, for each variable, the likelihood of the valued observed by assuming the Normal distribution.

In practice, the sum of the logarithms better suits the purpose of this work, since the final probability may be so small that can be affected by the limitations of floating-point arithmetic. Additionally, both PAT and ODIn only care about the order of the values, rather than their magnitude, and the exponential function is monotonic. Figure 53 illustrates the behavior of scores obtained with ONB.

Figure 53 – Illustration of a typical density function of one-class scores derived from One-class Naive Bayes. The positive class is female *Aedes aegypti*. One negative sub-class (male *Aedes aegypti*) was included only to give a sense of relative scale to the figure. Dataset: Insects v2, described in Section 5.3.6.

Source: Research data.

The main disadvantage of ONB over Mahalanobis and OSVM is that ONB assumes the features to be independent of one another, whereas this is not the case with the other scorers. One advantage is that, given large enough b , ONB can model density functions that are not clustered within either hyperspherical or hyperelliptical shapes. Mahalanobis, on the other hand, always assumes a hyperspherical grouping of data, and while OSVM can model non-hyperspherical shapes, choosing an appropriate kernel is needed.

We particularly designed ONB to fit the characteristics of the dataset Insects v2, as it reflects our motivating application. In that case, we compare the performance of ONB against other one-class scorers in terms of AUC. The number of bins in ONB is arbitrarily set to 30 and was not tuned. Our empirical results are shown in Table 19. ONB produced the highest AUC for every negative class in this dataset when the positive class is fixed as female *Aedes aegypti*.

Table 19 – AUC produced by several one-class scorers and varying negative classes, where the positive class is female *Aedes aegypti*. Dataset: Insects v2 (see Section 5.3.6).

label	LOF	OSVM _{scale}	OSVM _{auto}	MD	IF	ONB
♂ European bee	0.28	0.66	0.72	0.98	0.98	0.99
♂ <i>Aedes aegypti</i>	0.87	0.93	0.72	0.98	0.98	0.98
♀ <i>Anopheles</i>	0.75	0.25	0.72	0.74	0.83	0.96
♂ <i>Anopheles</i>	0.70	0.79	0.71	0.77	0.78	0.83
♂ <i>Chironomos</i>	0.55	0.50	0.71	0.76	0.77	0.79
♂ <i>Cotesia</i>	0.23	0.55	0.72	0.96	0.98	1.00
♂ <i>Crisopideo</i>	0.27	0.54	0.72	0.97	0.98	0.99
♀ <i>Culex quinquefasciatus</i>	0.71	0.19	0.69	0.42	0.53	0.74
♂ <i>Culex quinquefasciatus</i>	0.81	0.87	0.72	0.92	0.94	0.95
♂ <i>Diatraea</i>	0.40	0.41	0.72	0.97	0.98	0.99
♂ <i>Diatraea</i>	0.47	0.39	0.72	0.99	0.98	0.99
♀ Fruit fly	0.31	0.15	0.72	0.96	0.96	0.99
♂ Fruit fly	0.37	0.12	0.72	0.96	0.95	0.99
♂ <i>Melonogaster</i>	0.29	0.17	0.72	0.96	0.96	0.99
♂ <i>Musca domestica</i>	0.27	0.52	0.72	0.96	0.98	0.99
♂ <i>Scaptotrigona bipunctata</i>	0.28	0.73	0.72	0.97	0.99	0.99
♂ <i>Tetragonisca angustula</i>	0.24	0.65	0.72	0.98	0.98	0.99

Source: Research data.

5.5.2 Random Forest for c Estimation

With ExTlce, we demonstrated that Tlce is heavily impacted by the bias in its greedy search. In this section, we further investigate the algorithm and properly test our hypothesis that the Tlce’s correction δ_γ is only necessary due to the heavy bias in the method towards region of the feature space that has unusually elevated $\hat{c}$.

To this end, we developed a baseline method that we call Random Forest for c Estimation (RanFocE). Consider the minimum node size l . Each tree T has its nodes split randomly: for each node, one feature that can split its corresponding data into two sets with *at least* l labeled data points is chosen, and the data is split according to said feature at a random threshold value under the same constraint. If no such a feature exists, the node is split no more. The estimation of c for a node n is $\hat{c}_n$ and is the total number of labeled data points divided by the total number of data points. The estimation of c for a tree is $\hat{c}_T = \max_{n \in T} \hat{c}_n$. The final estimation is the median of the estimates provided by all trees in the forest. We note that, contrary to Tlce, no correction is applied to the estimations. We emphasize the absence of any correction factor whatsoever.

In our experiments, we induced 100 trees per forest. We adopted a value for l similar to the minimum number of data points in Tlce: $l = \min\{1000, \lfloor 0.5 + 0.1 \times |L| \rfloor\}$.

Our empirical results under the settings proposed by Experiment #3-a and Experiment #3-c, which are described in Section 5.3, are shown in Table 20. With this data, we cannot infer statistical difference between the algorithms in either setting (p -values of 0.36 and 0.41 for experiments #3-a and #3-c, respectively, according to a Wilcoxon Rank-Test). Therefore, we cannot conclude that Tlce performs better than a similar tree induction algorithm that splits randomly and do not perform a correction to the final

estimation of c .

Table 20 – Mean absolute error (standard deviation in parentheses) of TIcE and RanFocE. Results are presented in percentages and include for experiments #3-a and #3-c.

Data	Experiment #3-a		Experiment #3-c	
	TIcE	RanFocE	TIcE	RanFocE
N	7.93 (1.27)	8.25 (1.15)	17.18 (3.17)	14.62 (2.09)
I	10.92 (1.42)	5.96 (1.38)	22.16 (3.02)	9.76 (2.98)
A	10.81 (3.59)	10.14 (1.99)	15.03 (4.01)	10.97 (1.64)
B	11.22 (2.12)	6.04 (1.11)	17.48 (2.76)	9.09 (1.81)
C	11.32 (2.07)	6.18 (1.31)	11.32 (2.07)	6.18 (1.31)
H	11.12 (1.97)	7.16 (1.27)	11.12 (1.97)	7.16 (1.27)
L	6.53 (2.86)	12.63 (2.24)	9.10 (3.59)	13.69 (2.28)
P	7.84 (3.18)	11.25 (1.99)	11.74 (3.75)	12.20 (2.12)
R	7.86 (2.13)	9.54 (2.12)	7.86 (2.13)	9.54 (2.12)
W	8.56 (1.97)	8.23 (1.42)	8.56 (1.97)	8.23 (1.42)
rank	1.6	1.4	1.7	1.3

Source: Research data.

5.6 Discussion

In this chapter, we concluded our third contribution listed in [Section 1.1](#). Our final discussion regarding this topic follows.

Elkan’s method (EN) has historical value as it put the spotlight on Positive and Unlabeled Prior Estimation, a problem that is similar to One-class Quantification. EN also introduced theoretical base for newer algorithms to improve on. However, for practical use, it performs generally poorly and we cannot recommend it. Instead, we argue that EN can and should be used as a baseline to be compared against.

We cannot recommend PE since it was not significantly better than EN and, at the same time, was shown to be the slowest approach among all algorithms tested.

We note that BFT represents the best possible Classify And Count derived from a one-class scorer. However, given the systemic error of CC, BFT not only can be surpassed but also should. We advocate that BFT should be viewed as a baseline for future approaches.

ExTlce fulfilled its role of showing the potential of Tlce’s underlying search problem. Indeed, the former consistently provided smaller absolute quantification errors than the latter. Nevertheless, our purpose is not to defend ExTlce’s position and recommend it as a quantifier, but rather entice the community to further explore the region search problem proposed by Tlce, in future work. ExTlce, while less restricted than Tlce, is still limited in a number of ways. For instance, like in Tlce and most other tree algorithms, the sub-regions explored only “cut” the feature space along its axes. Additionally, we believe it is possible to create an algorithm from the ideas of Tlce that, similarly to PAT, is capable of inducing a model that can later be used to quantify several test samples without resorting to the training data.

[Ramaswamy, Scott and Tewari \(2016\)](#) make the argument that “requiring an accurate conditional probability estimate (which is a real valued function over the feature space) for estimating the mixture proportion (a single number) is too roundabout”. On the other hand, we defend that the referred approach is actually very practical, since there are already a number of methods for this exactly purpose that are accessible for even inexperienced practitioners. This approach is also the base of PAT, which is, in our opinion, notoriously simpler than KM, yet generally providing smaller quantification errors at an unquestionable faster rate.

In our experiments, PAT was shown to produce the smallest quantification errors while being the fastest algorithm. For this reason, it is the algorithm we mostly recommend for practical use.

Notwithstanding the favorable results, we must highlight PAT’s drawbacks, which

were evidenced by the evolution of Experiment #3. PAT was developed on the assumption that some negative observations can be similar to positive observations up to a certain degree. The algorithm (indirectly) tries to ignore the presence of negative observations close to the boundaries of the positive class, in the feature space, by extrapolating the number of observations from only the top scored ones.

However, consider the case where a negative sub-class is partially identical to the positive class, in the sense that a *number of* negative observations are, *individually, identical to* or *indistinguishable from* positive observations. In such a case, the quantification of PAT will likely be affected, since PAT does its computations solely on the observations' scores. Naturally, the degree to which PAT will be affected depends on the proportion of the aforementioned sub-class within the negative class.

Meanwhile, ExTicE could be less or not affected by those partially identical classes. Indeed, its search mechanism allows it to completely ignore regions of the feature space where such overlaps are more prevalent, *if* there are other regions with less overlap. Figure 54 illustrates this discussion. Notice that ExTicE would likely only consider the top-right quadrant of the feature space to infer c , while PAT would use all scores, even though negative observations are as highly scored as positive observations, in this scenario.

Figure 54 – A hypothetical situation where the main approach of TicE would lead to better quantification than PAT's approach, regardless of the underlying scorer of the latter. Blue + (plus) symbol indicates positive observations, while red – (minus) symbol indicates negative observations.

Source: Elaborated by the author.

Remarks being done for PAT and ExTicE, we argue that, for practical reasons, the overlaps mentioned may indicate a need to revise: **(a)** whether the negative observations actually *should* or *need* be classified as negative and **(b)** the quality of the existing features.

In any case, we can try to minimize the effects of negative classes that are identical to positive observations on PAT by ensembling it along with ExTicE. In this particular scenario, PAT overestimates $\hat{p}$ as a result of negative observations being considered to be positive. In addition to that, we noticed that ExTicE tends to generally overestimate $\hat{p}$. The latter finding is not straightforward: since ExTicE biasedly tries to maximize $\hat{c}_\gamma$ (the proportion of labeled data over all positive data), we would expect $\hat{p}$ (the proportion of unlabeled positive data over all unlabeled data) to be overestimated. However, ExTicE overestimated 75% of its predictions in Experiment #3 (considering all classes). Such an overestimation can be justified by TicE’s correction factor δ_γ being too heavy. The occasional heavy overestimation of PAT along with the general overestimation of ExTicE favor the approach of considering the minimum between the predictions provided by both methods.

Table 21 – Mean absolute error (standard deviation in parentheses) of ExTicE, PAT_M , and an ensemble with both methods. Results are presented in percentages and include for experiments #3-a and #3-c.

Data	Experiment #3-a			Experiment #3-c		
	ExTicE	PAT_M	$\min\{\hat{p}\}$	ExTicE	PAT_M	$\min\{\hat{p}\}$
N	3.53 (1.81)	2.09 (1.75)	2.31 (1.69)	7.69 (3.24)	49.95 (0.76)	7.59 (3.19)
I	5.23 (2.35)	2.16 (1.67)	2.17 (1.70)	12.67 (3.22)	38.28 (4.42)	12.60 (2.94)
A	6.44 (2.28)	3.73 (2.69)	4.20 (2.32)	9.79 (2.95)	5.68 (3.67)	5.59 (2.93)
B	4.89 (2.17)	3.21 (1.99)	2.86 (1.76)	9.10 (2.81)	16.59 (3.57)	8.12 (2.45)
C	4.26 (2.30)	2.03 (1.63)	2.12 (1.70)	4.26 (2.30)	2.03 (1.63)	2.12 (1.70)
H	5.49 (2.34)	3.34 (3.76)	4.44 (6.96)	5.49 (2.34)	3.34 (3.76)	4.44 (6.96)
L	3.73 (2.53)	3.07 (2.38)	3.26 (2.56)	5.83 (3.16)	3.65 (2.47)	3.54 (2.61)
P	4.62 (2.68)	2.58 (2.11)	3.43 (2.52)	6.65 (3.02)	2.85 (2.25)	3.59 (2.41)
R	4.17 (2.57)	2.64 (2.09)	2.81 (2.17)	4.17 (2.57)	2.64 (2.09)	2.81 (2.17)
W	3.23 (2.46)	2.23 (1.87)	2.67 (1.93)	3.23 (2.46)	2.23 (1.87)	2.67 (1.93)
rank	3.0	1.1	1.9	2.7	1.8	1.5

Source: Research data.

Table 21 presents the results of experiments #3-a and #3-c for ExTicE, PAT_M and an ensemble that outputs the minimum prediction between the two methods. In experiment #3-a (median), we can see that, for all but one dataset, the ensemble performs better than ExTicE and worse than PAT_M . In the exceptional case of dataset A, the ensemble performed better than the other methods. We note that in all but one dataset, the performance of the ensemble was numerically closer to the performance of PAT_M rather than ExTicE. On the other hand, in experiment #3-c, the ensemble has a better performance than PAT_M in multiple datasets. Differently from PAT_M , the ensemble could perform well in the problematic datasets N, I and B. However, we emphasize that this

ensemble imposes a high computational cost due to the use of ExTlCE. Our main purpose is to highlight that it is indeed possible to achieve performance similar to PAT's while handling the particular case where it cannot perform well. We expect other, faster, methods to be developed in future work.

Finally, both PAT and ExTlCE strongly depend on the assumption that the distribution of the positive class is the same in both training and test samples. Given their strategies, we safely presume that they would be severely affected in the event of the assumption being false.

CONCLUSION

In this thesis, we studied the effects of the non-stationarity of data in classification and quantification for two main tasks: context identification and one-class quantification.

The main challenges revolving context identification are identifying to which context a sample of data belong to, among several previously known, and to disregard the proportion of classes when defining a context, that is, a context should be defined solely by the behavior of each class individually.

In this regard, in [Section 4.3](#) we proposed XO-HDy, a method that can reliably identify the context a sample of data is subjected to without having direct access to the factors that cause such a context, given that we known all possible contexts beforehand. Additionally, XO-HDy can operate under prior probability shift and assess the current proportion of classes, which can be used to dynamically set a better decision threshold for the classification task. Therefore, XO-HDy fulfills the first two contributions listed in [Section 1.1](#).

XO-HDy originally relies on a quantification method called HDy which, in turn, relies on the Helliger Distance. We further demonstrated that such a quantification method can be replaced. In that regard, we developed SORD, a distribution distance function that obtained great performance for quantification. Finally, we developed and studied the properties of MoSS, a synthetic scorer that supports DySyn, another quantification method.

We can pinpoint two limitations of XO-HDy. The first limitation is its inability to identify when a sample of data does not belong to any known context. This issue can be circumvented by setting a threshold to the dissimilarity measured between the predicted context and the test data. [Maletzke et al. \(2018\)](#) provide pointers on how to define such a threshold based on empirical percentiles.

The second and main limitation is that XO-HDy only operates in binary settings,

that is, cases where there are only two classes involved. A straightforward opportunity for improvement lies on making a multi-class version of the method. However, the reality of our motivating application is that, although we are interested in counting one or just a few target classes, the intelligent sensor can face a number of classes that is too large to be reasonably compiled into a training data set. If we are interested in just counting the number of insects that belong to a target species, *anything* that is observed and registered by the sensor that does not belong to such a species should be ignored. That includes, *but is not limited to*, millions of other species (CHAPMAN; SIMPSON; DOUGLAS, 2013) of insects. We emphasize that nothing prevents the sensor from registering everything that is not an insect. Therefore, the negative class is effectively unpredictable.

While we, with our experiments regarding context identification, established that it is essential to identify context, we need not do it by way of examining the distribution of flight data. If we refrain from ditching additional sensors, as thermometer and barometer, we are able to identify the context insects are subjected to without analyzing their flight characteristics. In this setting, we could get context identification out of our way and focus on our more crucial task: counting the insects that belong to the target class when the negative class is unpredictable.

In this regard, in Section 5.2 we propose PAT and ExTICe, which conclusively outperform the state-of-the-art in counting positive observation when the negative class is unpredictable. Particularly, we show that PAT has the best overall quantification performance while consistently presenting the lowest computational cost by orders of magnitude. ExTICe, on the other hand, can address a specific case where PAT fails, although ExTICe presenting overall lower quantification performance and exceedingly higher computational cost. PAT and ExTICe fulfills our third contribution listed in Section 1.1.

PAT relies on a one-class scorer. In Section 5.5.1 we introduce the One-class Naive Bayes, which performs better (in terms of AUC) than the state-of-the-art for insect data, according to our experiments.

PAT has two main limitations. The first is its inability to deal with a specific case where ExTICe exceeds. This limitation is further discussed in Section 5.6. One straightforward possibility for future work is to develop methods that can perform as well as PAT in all other conditions and outperform it in this specific case.

The second limitation is that the positive class is supposed to have a stationary distribution. This limitation implies that the positive class cannot aggregate subclasses. For instance, we may be interested in counting two species of insects simultaneously. This limitation can also be addressed in future work.

The main contributions of this thesis and additional contributions that had our collaboration were included in the following papers:

- **Title** Quantification in Data Streams: Initial Results;

Authors Maletzke, André and dos Reis, Denis and Batista, Gustavo;

Status published in the proceedings of Brazilian Conference on Intelligent Systems in 2017;

Description initial results of our team’s research on quantification applied to data streams;

My contributions revised the material to check for errors and mistakes, and helped improve writing;
- **Title** Unsupervised Context Switch for Classification Tasks on Data Streams with Recurrent Concepts;

Authors dos Reis, Denis and Maletzke, André and Batista, Gustavo;

Status published in the proceedings of the 33rd ACM/SIGAPP Symposium On Applied Computing in 2018;

Description in this paper, we share our initial results that motivate the exploration of context identification. Our first experiments were based on choosing the context that minimized the Kolmogorov Smirnov statistic, a technique that is rather simplistic compared to the later proposed XO-HDy;

My contributions developed the proposed technique, designed and executed experimental evaluation, and was main responsible for discussion, writing, and planning of experiments;
- **Title** Classifying and Counting with Recurrent Contexts;

Authors dos Reis, Denis and Maletzke, André and Silva, Diego and Batista, Gustavo;

Status published in the proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining in 2018;

Description in this paper, we further investigated the task of context identification and proposed SMR-HDy and XO-HDy;

My contributions developed the algorithms SMR-HDy and XO-HDy, designed and executed experimental evaluation, and was main responsible for discussion, writing, and planning of experiments;
- **Title** One-Class Quantification;

Authors dos Reis, Denis and Maletzke, André and Cherman, Everton and Batista, Gustavo;

Status published in the proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases in 2018;

Description in this paper, we proposed the task of One-class Quantification in addition to the methods ODIn and PAT. We compared said methods against topline and baseline algorithms that represented static *classify and count* approaches. We additionally investigated the behavior of widely used quantification methods when the availability of negative sub-classes vary in both training and test sets;

My contributions developed the algorithms PAT and ODIn, designed and executed experimental evaluation, and was main responsible for discussion, writing, and planning of experiments;

- **Title** On the Need of Class Ratio Insensitive Drift Tests for Data Streams;

Authors Maletzke, André and dos Reis, Denis and Cherman, Everton and Batista, Gustavo;

Status published in the proceedings of the Second International Workshop on Learning with Imbalanced Domains: Theory and Applications in 2018;

Description in this paper, we explored the use of algorithm HDy to unsupervisedly detect concept drifts while disregarding changes in the proportion of classes. We set a threshold to the Hellinger Distance based on percentiles learnt with validation data;

My contributions since this paper is a follow up of my previous work ([REIS et al., 2018b](#)) and addressed one of its main limitations (inability to identify previously unseen context), I was able to help design the proposed solution, plan experiments, and discuss results. I also helped improve the writing and checked for errors;

- **Title** Combining instance selection and self-training to improve data stream quantification;

Authors Maletzke, André and dos Reis, Denis M and Batista, Gustavo;

Status published in the Journal of the Brazilian Computer Society in 2018;

Description in this paper, explore the use of Kolmogorov-Smirnov test to detect drifts in quantification tasks. We further investigate instance selection and self-training to avoid requesting true labels to retrain the quantification model after a drift is detected;

My contributions since this paper share similarities with my previous work ([REIS et al., 2016](#)), I was able to help plan experiments, check for possible mistakes, and discuss results. I also helped improve the writing;

- **Title** DyS: a Framework for Mixture Models in Quantification;

Authors Maletzke, André and dos Reis, Denis and Cherman, Everton and Batista, Gustavo;

Status published in the proceedings of the Thirty-Three AAAI Conference on Artificial Intelligence in 2019;

Description in this paper, we introduce DyS, a quantification algorithm that generalizes HDy, and investigate the adoption of several distribution dissimilarity functions. Additionally, we introduce Mixable KS (MKS) and SORD, two dissimilarity functions for univariate distributions. SORD is described in [Section 4.6.1](#);

My contributions developed MKS and SORD, which are part of the main contributions, and was responsible for describing them. I also helped plan experimental evaluation, discussed results with the other authors, checked for errors and helped improve the writing;

- **Title** Challenges in Benchmarking Stream Learning Algorithms with Real-world Data;

Authors Souza, Vinícius and dos Reis, Denis and Maletzke, André and Batista, Gustavo;

Status submitted to Data Mining and Knowledge Discovery in 2019 and under second round of revision;

Description in this paper, we discuss the main characteristics of data streams and how they occur in widely used benchmark datasets. We emphasize the limitations of said data and provide new datasets that cover several types of changes;

My contributions was the main responsible for the literature review about the probabilistic perspective on changes in data, suggested new categorization of datasets according to the nature of their observations, helped analyze the datasets we discussed, performed experiments related to temporal overlap, modified MOA to retrieve metrics associated with ADWIN, and helped improve the writing;

- **Title** Accurately Quantifying under Score Variability;

Authors Maletzke, André and dos Reis, Denis and Hassan, Waqar and Batista, Gustavo;

Status submitted to the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020;

Description in this paper, we propose DySyn, a method based on DyS that adopts synthetic scores rather than scores obtained with a validation set to model the

distribution of the classes. The synthetic scores follow our proposed model MoSS, described in [Section 4.6.3](#);

My contributions developed and analyzed properties of MOSS, which is part of the main contributions, and was responsible for describing it. I also helped plan experimental evaluation, discussed results with the other authors, checked for errors and helped improve the writing;

- **Title** The importance of the test set size in quantification assessment;

Authors Maletzke, André and Hassan, Waqar and dos Reis, Denis and Batista, Gustavo;

Status submitted to the International Joint Conference on Artificial Intelligence, 2020;

Description in this paper, we investigate the effects of varying the number of data points in test samples for quantification tasks, and confirm that the ranking of quantification algorithms is subject to change. We also propose a meta learning scheme to automatically choose the best suited quantification algorithm, given a test sample;

My contributions wrote the analytical analysis of the performance of Classify and Count, helped plan experimental evaluation, discussed results with the other authors, checked for errors and helped improve the writing in general;

- **Title** Quantifying With Only Positive Training Data;

Authors dos Reis, Denis and de Souto, Marcílio and de Sousa, Elaine and Batista, Gustavo;

Status pending submission to the Machine Learning Journal in 2020;

Description in this paper, we extend our previous work ([REIS *et al.*, 2018a](#)) by further comparing PAT against Positive and Unlabeled Learning (PU L) methods and improving Tlce, a PU L algorithm;

My contributions this is a direct extension paper of my previous work ([REIS *et al.*, 2018a](#)), therefore, in addition to all my previous responsibilities in the original paper, I improved the literature review on PU PE and critically analyzed each algorithm in the state-of-the-art and developed ExTlce.

BIBLIOGRAPHY

AL-OTAIBI, R.; PRUDÊNCIO, R. B.; KULL, M.; FLACH, P. Versatile decision trees for learning over multiple contexts. In: **Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases**. [S.l.: s.n.], 2015. p. 184–199. Citation on page 86.

ALIMOGLU, F.; ALPAYDIN, E.; DENIZHAN, Y. Combining multiple classifiers for pen-based handwritten digit recognition. Citeseer, 1996. Citation on page 142.

AMIR, M.; TOSHNIWAL, D. Instance-based classification of streaming data using emerging patterns. In: SPRINGER. **International Conference on Advances in Information and Communication Technologies**. [S.l.], 2010. p. 228–236. Citation on page 84.

BAKTASHMOTLAGH, M.; HARANDI, M.; SALZMANN, M. Distribution-matching embedding for visual domain adaptation. **The Journal of Machine Learning Research**, JMLR. org, v. 17, n. 1, p. 3760–3789, 2016. Citation on page 86.

BATISTA, G. E. A. P. A.; HAO, Y.; KEOGH, E.; NETO, A. M. Towards automatic classification on flying insects using inexpensive sensors. In: **Proceedings of Tenth International Conference on Machine Learning and Applications**. Honolulu, Hawaii, USA: IEEE Computer Society, 2011. v. 1, p. 364–369. ISBN 978-0-7695-4607-0. Citations on pages 38 and 40.

BEKKER, J.; DAVIS, J. Estimating the class prior in positive and unlabeled data through decision tree induction. In: **Thirty-Second AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2018. Citations on pages 30, 77, 78, 119, 121, 123, 124, 137, 139, 173, and 174.

BEN-DAVID, S.; BLITZER, J.; CRAMMER, K.; PEREIRA, F. Analysis of representations for domain adaptation. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2007. p. 137–144. Citations on pages 29 and 68.

BIFET, A. Adaptive learning and mining for data streams and frequent patterns. **SIGKDD Explorations Newsletter**, ACM, New York, NY, USA, v. 11, n. 1, p. 55–56, Nov. 2009. ISSN 1931-0145. Available: <<http://doi.acm.org/10.1145/1656274.1656287>>. Citation on page 31.

BIFET, A.; GAVALDA, R. Learning from time-changing data with adaptive windowing. In: SIAM. **Proceedings of the 2007 SIAM international conference on data mining**. [S.l.], 2007. p. 443–448. Citations on pages 31 and 81.

_____. Learning from time-changing data with adaptive windowing. In: SIAM. **Proceedings of the 2007 SIAM international conference on data mining**. [S.l.], 2007. p. 443–448. Citation on page 60.

BIFET, A.; GAVALDÀ, R. Adaptive learning from evolving data streams. In: SPRINGER. **International Symposium on Intelligent Data Analysis**. [S.l.], 2009. p. 249–260. Citations on pages 31 and 81.

BIFET, A.; HOLMES, G.; KIRKBY, R.; PFAHRINGER, B. Moa: Massive online analysis. **J. Mach. Learn. Res.**, JMLR.org, v. 11, p. 1601–1604, Aug. 2010. ISSN 1532-4435. Available: <<http://dl.acm.org/citation.cfm?id=1756006.1859903>>. Citation on page 32.

BIFET, A.; HOLMES, G.; PFAHRINGER, B. Leveraging bagging for evolving data streams. In: **Joint European conference on machine learning and knowledge discovery in databases**. [S.l.: s.n.], 2010. p. 135–150. Citations on pages 31 and 81.

BIFET, A.; READ, J.; ZLIOBAITE, I.; PFAHRINGER, B.; HOLMES, G. Pitfalls in benchmarking data stream classification and how to avoid them. In: **Joint European Conference on Machine Learning and Knowledge Discovery in Databases**. [S.l.: s.n.], 2013. p. 465–479. Citation on page 56.

BREUNIG, M. M.; KRIEGEL, H.-P.; NG, R. T.; SANDER, J. Lof: identifying density-based local outliers. In: ACM. **ACM sigmod record**. [S.l.], 2000. v. 29, p. 93–104. Citation on page 127.

CHA, S.-H. Comprehensive survey on distance/similarity measures between probability density functions. **International Journal of Mathematical Models and Methods in Applied Sciences**, v. 1, n. 4, p. 300–307, 2007. Citation on page 89.

_____. Taxonomy of nominal type histogram distance measures. In: **Proceedings of the American Conference on Applied Mathematics**. [S.l.: s.n.], 2008. Citations on pages 89 and 91.

CHA, S.-H.; SRIHARI, S. N. On measuring the distance between histograms. **Pattern Recognition**, Elsevier, v. 35, n. 6, p. 1355–1370, 2002. Citations on pages 89 and 105.

CHAN, Y. S.; NG, H. T. Estimating class priors in domain adaptation for word sense disambiguation. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics**. [S.l.], 2006. p. 89–96. Citation on page 69.

CHAPMAN, R.; SIMPSON, S.; DOUGLAS, A. **The Insects: Structure and Function**. [S.l.]: Cambridge University Press, 2013. ISBN 9780521113892. Citations on pages 115 and 156.

CHEN, Y.; WHY, A.; BATISTA, G. E.; MAFRA-NETO, A.; KEOGH, E. Flying Insect Classification with Inexpensive Sensors. **Journal of Insect Behavior**, v. 27, n. 5, p. 657–677, 2014. ISSN 08927553. Citation on page 69.

CHRISTOFFEL, M.; NIU, G.; SUGIYAMA, M. Class-prior estimation for learning from positive and unlabeled data. In: **Asian Conference on Machine Learning**. [S.l.: s.n.], 2016. p. 221–236. Citation on page 119.

CLAESEN, M.; SMET, F. D.; GILLARD, P.; MATHIEU, C.; MOOR, B. D. Building classifiers to predict the start of glucose-lowering pharmacotherapy using belgian health expenditure data. **arXiv preprint arXiv:1504.07389**, 2015. Citation on page 73.

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. Citations on pages 100 and 142.

DAVIS, P. J. Leonhard euler’s integral: A historical profile of the gamma function: In memoriam: Milton abramowitz. **The American Mathematical Monthly**, Taylor & Francis, v. 66, n. 10, p. 849–869, 1959. Citation on page 113.

DHEERU, D.; TANISKIDOU, E. K. **UCI Machine Learning Repository**. 2017. Available: <<http://archive.ics.uci.edu/ml>>. Citations on pages 29 and 53.

DIAZ, J. J.; COLONNA, J. G.; SOARES, R. B.; FIGUEIREDO, C. M.; NAKAMURA, E. F. Compressive sensing for efficiently collecting wildlife sounds with wireless sensor networks. In: **2012 21st International Conference on Computer Communications and Networks**. Munich: [s.n.], 2012. p. 1–7. Citations on pages 30 and 141.

DYER, K. B.; CAPO, R.; POLIKAR, R. Compose: A semisupervised learning framework for initially labeled nonstationary streaming data. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 25, n. 1, p. 12–26, 2014. Citations on pages 32, 64, 82, and 83.

ELKAN, C.; NOTO, K. Learning classifiers from only positive and unlabeled data. In: **ACM. Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.], 2008. p. 213–220. Citations on pages 30, 77, 78, 117, and 139.

FARIA, E. R.; GONÇALVES, I. J.; CARVALHO, A. C. de; GAMA, J. Novelty detection in data streams. **Artificial Intelligence Review**, Springer, v. 45, n. 2, p. 235–269, 2016. Citation on page 31.

FARIA, E. R. de; FERREIRA, A. C. P. de L.; GAMA, J. *et al.* Minas: multiclass learning algorithm for novelty detection in data streams. **Data Mining and Knowledge Discovery**, Springer, v. 30, n. 3, p. 640–680, 2016. Citations on pages 19, 32, 83, 98, 104, and 105.

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 62.

FEIGENBAUM, J.; KANNAN, S.; STRAUSS, M. J.; VISWANATHAN, M. An approximate l 1-difference algorithm for massive data streams. **SIAM Journal on Computing**, SIAM, v. 32, n. 1, p. 131–151, 2002. Citation on page 30.

FERNANDO, B.; HABRARD, A.; SEBBAN, M.; TUYTELAARS, T. Unsupervised visual domain adaptation using subspace alignment. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2013. p. 2960–2967. Citation on page 86.

FLACH, P. **Machine learning: the art and science of algorithms that make sense of data**. [S.l.]: Cambridge University Press, 2012. Citations on pages 68 and 111.

FORMAN, G. Counting positives accurately despite inaccurate classification. In: **SPRINGER. European Conference on Machine Learning**. [S.l.], 2005. p. 564–575. Citations on pages 49, 70, and 73.

_____. Quantifying trends accurately despite classifier error and class imbalance. In: **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. Philadelphia: [s.n.], 2006. p. 157–166. Citations on pages [131](#) and [132](#).

FREY, P. W.; SLATE, D. J. Letter recognition using holland-style adaptive classifiers. **Machine learning**, Springer, v. 6, n. 2, p. 161–182, 1991. Citation on page [141](#).

GAMA, J.; MEDAS, P.; CASTILLO, G.; RODRIGUES, P. Learning with drift detection. In: **Brazilian symposium on artificial intelligence**. [S.l.: s.n.], 2004. p. 286–295. Citations on pages [31](#) and [81](#).

GAMA, J.; RODRIGUES, P. P.; SPINOSA, E. J.; CARVALHO, A. C. P. L. F. de. **Knowledge discovery from data streams**. [S.l.]: Chapman & Hall/CRC Boca Raton, 2010. Citation on page [30](#).

GAMA, J.; ZLIOBAITE, I.; BIFET, A.; PECHENIZKIY, M.; BOUCHACHIA, A. A survey on concept drift adaptation. **ACM computing surveys**, ACM, v. 46, n. 4, p. 44, 2014. Citations on pages [57](#) and [58](#).

GOMES, H. M.; BARDDAL, J. P.; ENEMBRECK, F.; BIFET, A. A survey on ensemble learning for data stream classification. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 50, n. 2, p. 1–36, 2017. Citation on page [31](#).

GOMES, H. M.; BIFET, A.; READ, J.; BARDDAL, J. P.; ENEMBRECK, F.; PFHARINGER, B.; HOLMES, G.; ABDESSALEM, T. Adaptive random forests for evolving data stream classification. **Machine Learning**, Springer, v. 106, n. 9-10, p. 1469–1495, 2017. Citation on page [84](#).

GONZÁLEZ-CASTRO, V.; ALAIZ-RODRÍGUEZ, R.; ALEGRE, E. Class distribution estimation based on the hellinger distance. **Information Sciences**, Elsevier, v. 218, p. 146–164, 2013. Citation on page [87](#).

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 [30](#), [63](#), and [69](#).

GUHA, S.; MISHRA, N. Clustering data streams. In: **Data stream management**. [S.l.]: Springer, 2016. p. 169–187. Citation on page [31](#).

HAMMAMI, N.; BEDDA, M. Improved tree model for arabic speech recognition. In: **Proceedings of the International Conference on Computer Science and Information Technology**. [S.l.: s.n.], 2010. v. 5, p. 521–526. Citations on pages [100](#) and [141](#).

HAQUE, A.; KHAN, L.; BARON, M. Semi supervised adaptive framework for classifying evolving data stream. In: **Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining**. [S.l.: s.n.], 2015. p. 383–394. Citation on page [60](#).

HARRIES, M. **Splice-2 comparative evaluation: Electricity pricing**. Sydney, Australia, 1999. 1-55 p. Citations on pages [29](#) and [53](#).

- 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 31.
- 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 69.
- HUANG, D. T. J.; KOH, Y. S.; DOBBIE, G.; PEARS, R. Detecting volatility shift in data streams. In: **IEEE International Conference on Data Mining**. [S.l.: s.n.], 2014. p. 863–868. Citation on page 60.
- HUANG, J.; GRETTON, A.; BORGWARDT, K.; SCHÖLKOPF, B.; SMOLA, A. J. Correcting sample selection bias by unlabeled data. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2007. p. 601–608. Citation on page 86.
- IKONOMOVSKA, E.; GAMA, J.; DVEROSKI, S. Learning model trees from evolving data streams. **Data mining and knowledge discovery**, Springer, v. 23, n. 1, p. 128–168, 2011. Citation on page 54.
- JAIN, S.; WHITE, M.; RADIVOJAC, P. Estimating the class prior and posterior from noisy positives and unlabeled data. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2016. p. 2693–2701. Citations on pages 120 and 121.
- JAIN, S.; WHITE, M.; TROSSET, M. W.; RADIVOJAC, P. Nonparametric semi-supervised learning of class proportions. **arXiv preprint arXiv:1601.01944**, 2016. Citations on pages 30, 49, and 119.
- KATAKIS, I.; TSOUMAKAS, G.; BANOS, E.; BASSILIADES, N.; VLAHAVAS, I. An adaptive personalized news dissemination system. **Journal of Intelligent Information Systems**, Springer, v. 32, n. 2, p. 191–212, 2009. Citation on page 54.
- KELLY, M. G.; HAND, D. J.; ADAMS, N. M. The impact of changing populations on classifier performance. In: **Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining**. New York, NY, USA: ACM, 1999. (KDD '99), p. 367–371. ISBN 1-58113-143-7. Available: <<http://doi.acm.org/10.1145/312129.312285>>. Citation on page 66.
- KHAN, S. S.; MADDEN, M. G. A survey of recent trends in one class classification. In: SPRINGER. **Irish conference on artificial intelligence and cognitive science**. [S.l.], 2009. p. 188–197. Citation on page 126.
- 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, 84, and 98.
- KORYCKI, .; KRAWCZYK, B. Unsupervised drift detector ensembles for data stream mining. In: **Proceedings of the IEEE International Conference on Data Science and Advanced Analytics**. [S.l.: s.n.], 2019. Citations on pages 32, 85, and 98.
- KRAWCZYK, B.; MINKU, L. L.; GAMA, J.; STEFANOWSKI, J.; WOŹNIAK, M. Ensemble learning for data stream analysis: A survey. **Information Fusion**, Elsevier, v. 37, p. 132–156, 2017. Citation on page 31.

KULL, M.; FLACH, P. Patterns of dataset shift. In: **First International Workshop on Learning over Multiple Contexts at ECML-PKDD**. [S.l.: s.n.], 2014. p. 1–10. Citation on page 64.

KUMAGAI, A.; IWATA, T. Learning latest classifiers without additional labeled data. In: **International Joint Conferences on Artificial Intelligence**. [S.l.: s.n.], 2017. p. 2039–2045. Citations on pages 33 and 86.

_____. Unsupervised domain adaptation by matching distributions based on the maximum mean discrepancy via unilateral transformations. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2019. v. 33, p. 4106–4113. Citation on page 86.

KUNCHEVA, L. I.; SÁNCHEZ, J. S. Nearest neighbour classifiers for streaming data with delayed labelling. In: **IEEE International Conference on Data Mining**. [S.l.: s.n.], 2008. p. 869–874. Citation on page 84.

LICHMAN, M. **UCI Machine Learning Repository**. 2013. <<http://archive.ics.uci.edu/ml>>. Citations on pages 30, 100, 141, and 142.

LIU, F. T.; TING, K. M.; ZHOU, Z.-H. Isolation forest. In: IEEE. **2008 Eighth IEEE International Conference on Data Mining**. [S.l.], 2008. p. 413–422. Citation on page 129.

LONG, M.; CAO, Y.; WANG, J.; JORDAN, M. Learning transferable features with deep adaptation networks. p. 97–105, 2015. Citation on page 86.

LYON, R.; 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. Citation on page 142.

MAHALANOBIS, P. C. On the generalized distance in statistics. In: NATIONAL INSTITUTE OF SCIENCE OF INDIA. [S.l.], 1936. Citation on page 125.

MALETZKE, A. **Binary quantification in non-stationary scenarios**. Phd Thesis (PhD Thesis) — Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo, São Carlos, São Paulo, Brazil, 2019. Citation on page 109.

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, 109, 114, and 155.

_____. Dys: a framework for mixture models in quantification. In: **Thirty-Three AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2019. Citations on pages 81, 89, 99, 105, 107, 108, 109, and 135.

MARRS, G. R.; HICKEY, R. J.; BLACK, M. M. The impact of latency on online classification learning with concept drift. In: SPRINGER. **International Conference on Knowledge Science, Engineering and Management**. [S.l.], 2010. p. 459–469. Citation on page 82.

MASUD, M.; GAO, J.; KHAN, L.; HAN, J.; THURAISINGHAM, B. M. Classification and novel class detection in concept-drifting data streams under time constraints. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 23, n. 6, p. 859–874, 2010. Citation on page [83](#).

MASUD, M. M.; WOOLAM, C.; GAO, J.; KHAN, L.; HAN, J.; HAMLEN, K. W.; OZA, N. C. Facing the reality of data stream classification: coping with scarcity of labeled data. **Knowledge And Information Systems**, v. 33, n. 1, p. 213–244, 2012. Citation on page [83](#).

MELLANBY, K. Humidity and insect metabolism. **Nature**, v. 138, p. 124–125, 1936. Citation on page [89](#).

MORENO-TORRES, J. G.; RAEDER, T.; ALAIZ-RODRÍGUEZ, R.; CHAWLA, N. V.; HERRERA, F. A unifying view on dataset shift in classification. **Pattern Recognition**, Elsevier, v. 45, n. 1, p. 521–530, 2012. Citations on pages [30](#), [62](#), [63](#), and [64](#).

NGUYEN, H.-L.; WOON, Y.-K.; NG, W.-K. A survey on data stream clustering and classification. **Knowledge and information systems**, Springer, v. 45, n. 3, p. 535–569, 2015. Citation on page [31](#).

NOUMIR, Z.; HONEINE, P.; RICHARD, C. On simple one-class classification methods. In: IEEE. **2012 IEEE International Symposium on Information Theory Proceedings**. [S.l.], 2012. p. 2022–2026. Citation on page [126](#).

OZA, N. C. Online bagging and boosting. In: IEEE. **Proceedings of the 2005 IEEE international conference on systems, man and cybernetics**. [S.l.], 2005. v. 3, p. 2340–2345. Citations on pages [31](#) and [81](#).

PAN, S. J.; YANG, Q. A survey on transfer learning. **IEEE Transactions on knowledge and data engineering**, IEEE, v. 22, n. 10, p. 1345–1359, 2009. Citations on pages [29](#) and [68](#).

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011. Version: 0.20.3. Citations on pages [127](#) and [139](#).

PLESSIS, M. C. D.; SUGIYAMA, M. Class prior estimation from positive and unlabeled data. **IEICE TRANSACTIONS on Information and Systems**, The Institute of Electronics, Information and Communication Engineers, v. 97, n. 5, p. 1358–1362, 2014. Citations on pages [30](#), [119](#), and [139](#).

POZZOLO, A. D.; BORACCHI, G.; CAELEN, O.; ALIPPI, C.; BONTEMPI, G. Credit card fraud detection and concept-drift adaptation with delayed supervised information. In: **Proceedings of the International Joint Conference on Neural Networks**. [S.l.: s.n.], 2015. Citation on page [83](#).

QUIONERO-CANDELA, J.; SUGIYAMA, M.; SCHWAIGHOFER, A.; LAWRENCE, N. D. **Dataset shift in machine learning**. [S.l.]: The MIT Press, 2009. Citations on pages [29](#) and [68](#).

RAMASWAMY, H.; SCOTT, C.; TEWARI, A. Mixture proportion estimation via kernel embeddings of distributions. In: **International Conference on Machine Learning**. [S.l.: s.n.], 2016. p. 2052–2060. Citations on pages 30, 121, 139, and 151.

REIS, D. do. **Classificação de fluxos de dados com mudança de conceito e latência de verificação**. Master's Thesis (Master's Thesis) — Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo, São Carlos, São Paulo, Brazil, 2016. Citation on page 32.

REIS, D. dos; FLACH, P.; MATWIN, S.; BATISTA, G. Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. In: **ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [S.l.]: ACM, 2016. v. 22, p. 1545–1554. Citations on pages 32, 54, 85, 98, and 158.

REIS, D. dos; MALETZKE, A.; BATISTA, G. Unsupervised context switch for classification tasks on data streams with recurrent concepts. In: **SAC 2018 – The 33rd ACM/SIGAPP Symposium On Applied Computing**. [S.l.: s.n.], 2018. p. 518–524. Citations on pages 81, 89, 90, 98, 99, 100, and 141.

REIS, D. dos; MALETZKE, A.; CHERMAN, E.; BATISTA, G. One-class quantification. In: SPRINGER. **Joint European Conference on Machine Learning and Knowledge Discovery in Databases**. [S.l.], 2018. p. 273–289. Citations on pages 51, 71, 73, 115, 116, 129, 132, 133, 134, 139, 140, 141, and 160.

REIS, D. dos; MALETZKE, A.; SILVA, D. F.; BATISTA, G. E. Classifying and counting with recurrent contexts. In: ACM. **Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. [S.l.], 2018. p. 1983–1992. Citations on pages 38, 39, 49, 59, 64, 81, 95, 101, 102, 103, 104, 105, and 158.

REIS, D. dos; SOUTO, M. de; SOUSA, E. de; BATISTA, G. **Online supplemental material**. 2019. <<https://github.com/denismr/PUandOCQ>>. Citation on page 137.

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 105.

SAENKO, K.; KULIS, B.; FRITZ, M.; DARRELL, T. Adapting visual category models to new domains. In: SPRINGER. **European conference on computer vision**. [S.l.], 2010. p. 213–226. Citations on pages 29 and 68.

Attribute-value learning. In: SAMMUT, C.; WEBB, G. I. (Ed.). **Encyclopedia of Machine Learning**. Boston, MA: Springer US, 2010. p. 53–53. ISBN 978-0-387-30164-8. Available: <https://doi.org/10.1007/978-0-387-30164-8_43>. Citation on page 51.

SARNELLE, J.; SANCHEZ, A.; CAPO, R.; HAAS, J.; POLIKAR, R. Quantifying the limited and gradual concept drift assumption. In: **Proceedings of the International Joint Conference on Neural Networks**. [S.l.: s.n.], 2015. Citation on page 66.

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. Citation on page 37.

- SILVA, J. A.; FARIA, E. R.; BARROS, R. C.; HRUSCHKA, E. R.; CARVALHO, A. C. d.; GAMA, J. Data stream clustering: A survey. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 46, n. 1, p. 1–31, 2013. Citation on page 31.
- 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**, 2020. Citations on pages 51, 53, 58, 61, and 67.
- SOUZA, V. M.; SILVA, D. F.; GAMA, J. a.; BATISTA, G. E. Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: **SDM**. [S.l.]: SIAM, 2015. p. 873–881. Citations on pages 32, 82, and 83.
- SOUZA, V. M. A.; GIUSTI, R.; BATISTA, A. J. L. Asfalt: A low-cost system to evaluate pavement conditions in real-time using smartphones and machine learning. **Pervasive and Mobile Computing**, Elsevier, v. 51, p. 121–137, 2018. Citation on page 54.
- SOUZA, V. M. A.; SILVA, D. F.; BATISTA, G. E. A. P. A.; GAMA, J. Classification of evolving data streams with infinitely delayed labels. In: **International Conference on Machine Learning and Applications**. [S.l.: s.n.], 2015. p. 214–219. Citation on page 64.
- SOUZA, V. M. A.; SILVA, D. F.; GAMA, J.; BATISTA, G. Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: **SIAM International Conference on Data Mining**. [S.l.: s.n.], 2015. p. 873–881. Citations on pages 54 and 64.
- SUN, B.; SAENKO, K. Subspace distribution alignment for unsupervised domain adaptation. In: **Proceedings of The British Machine Vision Conference**. [S.l.: s.n.], 2015. v. 4, p. 24–1. Citation on page 86.
- SUREKA, A.; AGARWAL, S. Learning to classify hate and extremism promoting tweets. In: **IEEE. 2014 IEEE Joint Intelligence and Security Informatics Conference**. [S.l.], 2014. p. 320–320. Citation on page 73.
- TASCHE, D. Does quantification without adjustments work? **arXiv preprint arXiv:1602.08780**, 2016. Citation on page 72.
- 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 89.
- UPTON, G.; COOK, I. **A dictionary of statistics 3e**. [S.l.]: Oxford university press, 2014. Citation on page 53.
- VANSCHOREN, J.; RIJN, J. N. van; BISCHL, B.; TORGO, L. Openml: Networked science in machine learning. **SIGKDD Explorations**, ACM, New York, NY, USA, v. 15, n. 2, p. 49–60, 2013. Citation on page 141.
- WIDMER, G.; KUBAT, M. Learning in the presence of concept drift and hidden contexts. **Machine learning**, Springer, v. 23, n. 1, p. 69–101, 1996. Citation on page 31.
- 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 83.

XU, R.; CHEN, Z.; ZUO, W.; YAN, J.; LIN, L. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2018. p. 3964–3973. Citations on pages 33 and 86.

ZHAO, H.; ZHANG, S.; WU, G.; GORDON, G. J. *et al.* Multiple source domain adaptation with adversarial learning. In: **Proceedings of the International Conference on Learning Representations**. [S.l.: s.n.], 2018. Citations on pages 33 and 86.

ZHU, X. **Stream data mining repository**. 2010. Available: <www.cse.fau.edu/~xqzhu/stream.html>. Citations on pages 29 and 53.

ZHU, Y.; ZHUANG, F.; WANG, D. Aligning domain-specific distribution and classifier for cross-domain classification from multiple sources. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2019. v. 33, p. 5989–5996. Citations on pages 33, 86, and 87.

ŽLIOBAITĖ, I. Change with delayed labeling: when is it detectable? In: IEEE. **Proceedings of the International Conference on Data Mining Workshops**. [S.l.], 2010. p. 843–850. Citations on pages 32, 64, 84, and 98.

ZLIOBAITE, I. Combining similarity in time and space for training set formation under concept drift. **Intelligent Data Analysis**, IOS Press, v. 15, n. 4, p. 589–611, 2011. Citation on page 54.

ŽLIOBAITĖ, I.; BIFET, A.; READ, J.; PFAHRINGER, B.; HOLMES, G. Evaluation methods and decision theory for classification of streaming data with temporal dependence. **Machine Learning**, Springer, v. 98, n. 3, p. 455–482, 2015. Citation on page 56.

GLOSSARY

class overlap: The superposition between the distribution of two or more classes in the feature space..

class-feature: A special feature (unconsidered in the feature space) that denotes the category a data point belongs to..

class: One possible value that the class-feature can assume..

classifier: A model output by a learning algorithm that can be used to categorize data points..

concept drift/shift: Difference between the distributions of two sets of data points..

context: A delimitation of variables that are external to the feature space and nonetheless influence the distribution of data points in the feature space..

dataset shift: Same as concept drift..

deployment: The stage of an real-world application when mathematical models are put to use for practical purposes. After this point, the performance of models usually can no longer be validated..

feature space: Geometrical representation of the domain of a data point. Each dimension of the feature space represents one observable characteristic of a data point..

feature/attribute value: The observed characteristic of a data point. E.g. *green*, for the feature *prevalent color*..

feature/attribute: One single observable characteristic of a data point. E.g. *prevalent color*..

label latency/delay: The period of time between the instant when a data point is observed and the acquisition of its true label..

label or class-label: The value assigned to the class-feature of a data point..

learning algorithm: An algorithm that analyzes sets of observed data points and, through induction, outputs a mathematical model that aims to represent some aspect of the data, or support some specified task..

one-class scorer: A scorer that is induced with data points that belongs to the positive class exclusively..

quantifier: A model output by a learning algorithm that can be used to assess the prevalence of class-labels in a set of data points..

scorer: A model output by a learning algorithm that, given a data point, issues a value correlated with the probability of the data point belonging to the positive class..

test set/data: Set containing data points that are not used by the learning algorithm and that generally include all information needed to *test* some aspect of a model, such as its performance. For simplification, in this manuscript we loosely call test set any set that is unused as training data, included data collected after deployment. We note, though, that we typically are unable to test a model by using data collected after deployment..

training data/set: Set containing data points that are used to induce models through learning algorithms..

validation data/set: Set complementary to training data. It has other data points than the ones found in the training data but, for each point, it includes the same amount of information. Validation sets can be used as synthetic test sets (by opportunely hiding some information that the model aims to predict) or auxiliary sets to adjust parameters of the learning algorithm..

ONE-CLASS QUANTIFICATION

This chapter contains appendices related to [Chapter 5](#).

A.1 Analysis on TlcE's time complexity

In this section, we thoroughly analyze the time complexity of TlcE. For this analysis, consider only binary nominal attributes and splits at the median for numerical attributes, so that the data is always split into two slices.

To evaluate the goodness for each attribute, when splitting a node i , it is necessary to count how many positive observations go to each side after the split. For this end, the code provided by [Bekker and Davis \(2018\)](#) uses a data structure called *BitArray* for such a counting. The structure is instantiated and initialized for each possible split. Although *BitArray* is highly optimized, especially so for memory usage, it still performs the counting in $O(n_i)$, where n_i is the number of observations assessed by the splitting node. Additionally, any data structure that is below $O(n_i)$ for exact counting would still have an initialization that is $\Omega(n_i)$, since every observation must be processed to give enough information to the structure about the counting. $O(n_i)$ is the same time complexity of a linear counting using a standard array. The authors do not comment on alternatives.

We note that it is possible to use another data structure, like binary decision trees, to obtain the count in $O(\log n_i)$. This data structure can be updated after the split for the attribute that caused the split: for numerical attributes, this can be done with a Cartesian tree in $O(\log n_i)$, and for binary nominal attributes, it is not necessary since the attribute should not be used any longer. However, for the remaining attributes, there is no such a way to quickly place each observation into the correct side of the split, since no relation between the splitting attribute and the other ones is guaranteed. Therefore, the data structure for each attribute should be updated, resulting in a $O(m_i n \log n_i)$ for the

split when using such a data structure, where m_i is the number of attributes that the node has access to. On the other hand, by ditching this data structure, the split is $O(m_i n_i)$.

Considering that each attribute is used only once, the data is always split in half and *there is enough data*, the maximum height is m , *i.e.*, the total number of attributes, and the complexity of the algorithm is $O(2^m n)$, as shown in Equation (A.1), where n is the total number of observations, and $F(m, n)$ is the recurrence relation of the algorithm.

$$\begin{aligned}
 F(h, n) &= mn + 2F\left(m-1, \frac{n}{2}\right) \\
 &= mn + 2\left((m-1)\frac{n}{2} + 2F\left(m-2, \frac{n}{4}\right)\right) \\
 &= mn + 2(m-1)n + 4F\left(m-2, \frac{n}{4}\right) \\
 &\dots \\
 &= mn + 2(m-1)n + \dots + 2^{m-1}(m-m+1)n \\
 &= n \sum_{k=0}^m 2^k (m-k) \\
 &= 2n(-m + 2^m - 1) \\
 &O(2^m n)
 \end{aligned} \tag{A.1}$$

If there is not enough data to use all attributes, but, again, each attribute is used *only once*, and divide the data in half, the complexity is $O(n^2)$, since the maximum height is $\log_2 n$ and $2^{\log_2 n} = n$. Therefore, the general complexity of the algorithm is $O(\min(n^2, 2^m n))$ when each feature is used only once and the data is divided in half, which is significantly higher than the $O(mn)$ stated by Bekker and Davis (2018). If the attributes can be used more than once and/or the data is not evenly divided after each split, the complexity is even higher. This fact emphasizes the overly optimistic initial assessment of Bekker and Davis (2018).

A.2 Comparison between PAT and ODIn

In Table 22, we present results for the preliminary comparison between PAT and ODIn. We adopted the Mahalanobis distance for both methods, and therefore call them PAT_M and ODIn_M, respectively, to maintain consistence with the other experiments.

A.3 Exp #2

In Table 23 we present the results obtained in Experiment #2, which were omitted in Section 5.4.

Table 22 – Mean absolute error (standard deviation in parentheses), in percentages, for experiment #1 with only PAT_M and ODIn_M.

Data	PAT _M	ODIn _M	BFT _M
N	8.92 (2.43)	11.85 (1.70)	8.45 (1.49)
I	7.34 (2.48)	13.51 (1.87)	10.28 (1.71)
A	3.87 (2.79)	8.43 (2.70)	6.45 (2.40)
B	5.76 (2.37)	11.64 (2.09)	10.04 (1.81)
C	2.03 (1.63)	6.40 (2.00)	10.85 (3.77)
H	3.66 (4.69)	4.33 (2.42)	4.76 (3.32)
L	3.18 (2.33)	4.26 (1.78)	2.99 (1.66)
P	2.57 (1.98)	3.39 (1.38)	2.71 (1.47)
R	2.64 (2.09)	10.02 (3.16)	11.76 (3.42)
W	2.23 (1.87)	3.71 (1.68)	4.94 (2.18)
rank	1.2	2.6	2.2

Source: Research data.

Table 23 – Mean absolute error (standard deviation in parentheses), in percentages, for experiment #2.

Data	EN _a	EN _s	PE	KM1	KM2	TicE	ExTicE	PAT _M	BFT _M
N	27.90 (16.17)	14.55 (7.19)	17.46 (4.27)	7.51 (5.49)	15.41 (4.40)	16.90 (3.09)	8.73 (3.23)	7.11 (3.58)	8.38 (2.79)
I	26.77 (3.61)	25.65 (5.00)	21.20 (2.12)	12.42 (4.85)	13.92 (4.72)	22.46 (3.78)	13.16 (3.72)	8.50 (3.46)	10.13 (2.62)
A	12.38 (6.05)	12.26 (6.00)	14.91 (7.00)	24.51 (11.24)	26.02 (7.52)	20.84 (5.32)	12.80 (3.63)	4.55 (3.19)	5.78 (2.64)
B	15.40 (4.24)	12.26 (3.81)	17.15 (2.46)	14.81 (7.85)	18.80 (5.22)	17.95 (3.06)	9.24 (3.08)	6.70 (3.12)	10.58 (2.78)
C	14.86 (4.82)	12.64 (4.63)	11.79 (2.09)	12.02 (5.71)	15.25 (3.92)	11.32 (2.07)	4.26 (2.30)	2.03 (1.63)	10.77 (3.92)
H	8.42 (3.18)	49.30 (1.75)	12.29 (2.93)	–	–	10.96 (2.29)	5.62 (2.47)	3.23 (4.15)	5.50 (6.68)
L	11.75 (7.40)	14.05 (6.27)	15.05 (5.50)	15.84 (7.98)	17.40 (5.86)	13.02 (4.04)	8.09 (3.82)	5.32 (3.31)	3.13 (1.52)
P	50.82 (5.08)	10.82 (6.11)	11.05 (3.87)	12.26 (6.90)	19.33 (5.37)	13.90 (4.33)	7.05 (3.46)	2.92 (2.29)	2.29 (1.53)
R	24.19 (11.21)	14.46 (4.73)	15.44 (2.94)	6.46 (4.77)	10.47 (3.07)	7.86 (2.13)	4.17 (2.57)	2.64 (2.09)	11.64 (3.65)
W	12.12 (5.31)	21.98 (4.86)	16.20 (2.37)	5.45 (3.91)	8.77 (4.28)	8.56 (1.97)	3.23 (2.46)	2.23 (1.87)	5.01 (2.44)
rank	6.9	6.0	6.6	5.3	7.5	5.9	3.0	1.2	2.6

Source: Research data.