Universidade Estadual de Londrina

VINICIUS EIJI MARTINS

META-LEARNING FOR ACTIVE LEARNING TUNING ON STREAM CLASSIFICATION

Londrina
2022

VINICIUS EIJI MARTINS

META-LEARNING FOR ACTIVE LEARNING TUNING ON STREAM CLASSIFICATION

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

Supervisor: Prof. Dr. Sylvio Barbon Jr.

Londrina
2022

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Martins, Vinicius Eiji.
Meta-learning for Active Learning Tuning on Stream classification / Vinicius Eiji Martins. - Londrina, 2022.
65 f.

Orientador: Sylvio Barbon Jr.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2022.
Inclui bibliografia.

1. Meta-learning - Tese. 2. Active Learning - Tese. 3. Concept drift - Tese. 4. Stream Classification - Tese. I. Barbon Jr, Sylvio. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU 519

VINICIUS EIJI MARTINS

META-LEARNING FOR ACTIVE LEARNING TUNING ON STREAM CLASSIFICATION

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

_____

Orientador: Prof. Dr. Sylvio Barbon Jr.
Universidade Estadual de Londrina – UEL

_____

Prof. Dr. Bruno Bogaz Zarpelão
Universidade Estadual de Londrina – UEL

_____

Prof. Dr. Luiz Fernando Carvalho
Universidade Tecnológica Federal do Paraná,
Campus Apucarana – UTFPR

Londrina, 11 de março de 2022.

This thesis is dedicated to all those working towards a better world.

ACKNOWLEDGEMENTS

First I would like to thank my family, whose efforts and sacrifices afforded me the opportunities that allowed me to follow this path.

I would like to especially thank my supervisor, Sylvio Barbon Jr., who not only guided me through my undergraduate years, but also provided support, guidance, and inspiration through this master's program period with patience and dedication.

I also wish to extend my gratitude to the professors that accepted to evaluate this work, Professor Bruno Bogaz Zarpelão and Professor Rafael Gomes Mantovani.

I would also like to thank Professor Luiz Fernando Carvalho who was able to attend and evaluate this work under very short notice when Professor Rafael became unable to attend due to health complications.

I wish to thank my girlfriend, Ana Luisa, whose support was essential to get me through the obstacles that these pandemic years brought. Finally, I would like to extend my thanks to everyone that contributed to this work, be it directly or indirectly.

MARTINS, V. E.. **Aplicação de Meta-aprendizado para ajuste de Aprendizado Ativo sob tarefas de classificação de streams**. 2022. 65f. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2022.

RESUMO

Aprendizado de Máquina Supervisionado exige o conhecimento de todos os rótulos dos dados a serem utilizados para se treinar um modelo. Enquanto isso é factível em situações onde os dados são estáticos, em situações de dados contínuos, ou *streaming*, isso se torna inviável. Não só pela quantidade potencialmente infinita de dados, como também a velocidade exigida do modelo para que não se crie um atraso muito grande. Aprendizado Ativo (*Active Learning*) já resolveu uma porção considerável desse problema, diminuindo drasticamente o número de rótulos necessários para treinar um modelo enquanto se mantém uma performance competitiva. Essa técnica porém ainda apresenta seus desafios, um deles é o ajuste necessário da técnica com base na tarefa a ser realizada e a possibilidade de que os conceitos e comportamentos dos dados sendo analisados mudem, exigindo uma atualização não só do modelo sendo treinado, como também dos parâmetros de aprendizado ativo ajustados. Levando isso em conta, este trabalho busca reduzir esse problema com o uso de Meta-aprendizado. Utilizamos Aprendizado Ativo com amostragem baseada em *streaming* utilizando uma métrica de incerteza para decidir a necessidade de rotulação. Esta métrica é comparada com um valor de limiar chamado Z. Com meta-aprendizado podemos decidir o valor de Z dinamicamente e automaticamente, promovendo a adaptabilidade do modelo quanto a mudança de conceitos e eliminando o passo inicial de decidir qual valor de Z é mais propício para a tarefa a ser realizada. Nossos experimentos demonstraram uma redução na quantidade de rótulos necessários para o treinamento (média de 55.5%) com uma acurácia similar ao uso de *Very Fast Decision Tree* (VFDT) sob diferentes situações de mudança de conceitos em *streams*.

Palavras-chave: Meta-aprendizado. Aprendizado Ativo. Ajuste Dinâmico. Mudança de conceitos. Classificação de Streams.

MARTINS, V. E.. **Meta-learning for Active Learning Tuning on Stream classification**. 2022. 65p. Master's Thesis (Master in Science in Computer Science) – State University of Londrina, Londrina, 2022.

ABSTRACT

Supervised Machine Learning requires knowing the labels from the entire dataset being used for its training. While this is manageable in scenarios where the data is static, in situations where data is continuous, or in streaming situations, this becomes infeasible. Not only because of the potentially infinite amount of data, but also because of the speed necessary for the model so that a long delay is avoided. Active Learning has already solved a good portion of this problem, drastically lowering the number of required labels to train a model, all while maintaining competitive performance. However, this technique does present its own set of challenges. For example, the required tuning for each task may become outdated should the concepts and behaviours from the data change. This would require that not only a change be made to the model being trained, but also to the active learning parameters. Taking this into consideration, this work tackles this problem with the use of Meta-learning. We used Active Learning with Stream-based Selective Sampling and Uncertainty Sampling, a metric that uses a threshold value named Z. With Meta-learning we can set Z's value dynamically and automatically, providing adaptability to the model regarding concept drifts and also removing the initial step of choosing an optimal Z for the task at hand. Our experiments showed a reduction in the number of labels necessary for model training (average of 55.5%) with a similar performance to the use of Very Fast Decision Trees (VFDT) under different scenarios of concept drift in data streams.

Keywords: Meta-learning. Active Learning. Dynamic Tuning. Concept drift. Stream Classification.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

Z Uncertainty Sampling Threshold Value.

ActL Active Learning.

ADWIN ADaptive WINdowing.

CART Classification and Regression Tree.

HAT-ADWIN Hoeffding Adaptive Tree using ADWIN.

ML Machine Learning.

MLP Multilayer Perceptron.

MtL Meta-learning.

NN Neural Network.

ReLU Rectified Linear Unit Function.

RF Random Forest.

RMSE Root Mean Squared Error.

SVFDT Strict Very Fast Decision Tree.

SVM Support Vector Machine.

SVR Support Vector Regression.

TSFEL Time Series Feature Extraction Library.

VFDT Very Fast Decision Tree.

Z_MtL Dynamic threshold Active Learning.

CONTENTS

1 INTRODUCTION

Modern Supervised Machine Learning tasks require the extraction of hidden patterns from large amounts of data. These are usually fashioned as high speed, continuous data streams, which demand algorithms that can deal with a fast processing time and limited memory constraints [1, 2, 3].

When compared to the traditional batch based Supervised Machine Learning tasks, we can note the introduction of quite a few new challenges and constraints. In particular, the high cost of labelling samples and the occurrence of concept drift.

We can define a data stream as a potentially unbounded sequence of data samples $S =< S_1, S_2, ..., S_n, ... >$, where S_n is the n-th sample arrived. In the context of classification tasks, each sample can be seen as a d-dimensional feature vector $X =< x_1, x_2, ..., x_d >$ related to a class y that is usually unknown at the time of arrival. Since supervised learning tasks require the knowledge of the ground truth labels (y) to train a model, these samples must be labelled. Unfortunately, these samples are constantly arriving at high speeds as noted before, which drastically increases the cost of labelling.

The other challenge is the change in behaviour of new data, also referred to as concept drift. Concept drifts can be seen as the manifestation of a fundamental aspect in Data Streams, the concept of volatility. Examples of this phenomenon would be a sudden storm after days of sunny days in a weather monitoring system or a surge in power usage in an electrical grid when a component fails. Concept drift makes old models obsolete, requiring an adaptation to the new situation on their part.

All of this shows us that data stream tasks require a high level of consideration in regards to memory, time and volatility constraints. Constraints that are not as common in static batch-based tasks. But, with the emergence of Big Data and the Internet of Things (IoT) allowing for the generation of incredibly large amounts of data, real-world situations are almost always gravitating towards data streams. Big contributors to this are the lowering costs of storage and increasing Internet speeds and processing power found in electronic devices.

In regards to the cost of label acquisition, there exist several solutions that attempt to cope with its constraints. An example of such a solution would be the use of semi-supervised learning techniques to create micro-clusters based on labelled samples. This allows for the prediction of labels for samples that do not possess one through a k-NN classifier based on their position within a micro-cluster [4]. Among these solutions, many works tackle Active Learning allowing the model to learn from only a subset of samples [5, 6, 7, 8, 9].

Active Learning (ActL) is an area of machine learning that leans on the idea that a model can be trained faster and cheaper if it is allowed to choose what data it should use to learn. There exist techniques and uses of Active Learning for the most varied scenarios, but Selective Sampling is the most commonly used scenario [10].

Selective Sampling is a strategy that relies on a heuristic to decide whether a data sample is worth being used for training the model or not, querying an agent for the label of the sample if the former is decided. The most commonly used heuristic, or *query strategy*, is Uncertainty Sampling [11] which uses an uncertainty measure (H). This measure is usually the Entropy of the class distribution probabilities for the data sample according to the learning model. If $H(S_i) \geq Z$, where Z is a threshold value, then the data sample is queried and used in the training.

However, choosing an optimal Z poses a new challenge because its value is influenced by a variety of factors in the data being analysed. Factors such as the speed of arrival, the quantity, the learning algorithm, or the context of the data, among others. In a data stream scenario, all these changes can occur between different sections of the same stream thanks to concept drift. This makes a static Z not feasible for real-life scenarios.

One approach is to learn an initial Z and redefine it after a concept drift happens. Concept drift can be estimated by change detectors, algorithms capable of estimating the position of concept drifts in a data stream. They trigger an update in the base learner after changes in the data distribution to improve the classification task [12]. However, this still leaves the problem of choosing the optimal Z for each situation, a costly and slow process rendered unfeasible under streaming constraints.

We believe that using the performance of previous related tasks can help make a suitable recommendation of Z. Thus, we propose the use of Meta-learning (MtL) to automatically adjust the Z value of Uncertainly Sampling. The decisions are based on statistical meta-features extracted from an adaptive windowing procedure handling the trade-off between the number of labelling queries and high accuracy. We explored the Very Fast Decision Tree as a classification algorithm over different streams and concept drift conditions.

1.1 Objectives

The main objective of this work is to propose a framework that combines Meta-learning with Active Learning to better adapt models in stream settings, while also keeping a low amount of necessary samples for the model training.

Bearing this in mind, we define the specific goals as follows:

1. Propose a novel Active Learning framework that leverages Meta-learning techniques;

2. Compare the performance of this new framework with the use of traditional Active Learning techniques;
3. Compare the performance of the new framework with the use of traditional Supervised Learning;
4. Analyse the impact of concept drift in traditional Active Learning, Supervised Learning and the framework proposed in this work;

1.2 Contributions

The contributions brought by this work can be summarised as follows:

1. A new methodology to improve the stream classification performance under a reduced number of class labels, i.e., a real-life scenario;
2. The usage of Meta-learning for Active Learning tuning as a regression problem;
3. A discussion around the importance of different meta-feature categories in the context of entropy-based hyperparameter selection;
4. A discussion on how VFDT is impacted by concept drifts over different scenarios and how the classification performance may be improved when some data samples are ignored during training.

1.3 Outline

This master's thesis is organized as follows. In Chapter 2, it is provided the background related to machine learning and supervised learning, data streams and stream mining, active learning and meta-learning, all concepts used and combined in this work. Chapter 3 presents how this work relates to previous works in the literature. Chapter 4 describes how our approach is defined, the datasets used, and experimental details. Chapter 5 reports and discusses the results and insights acquired from our experiments. Finally, Chapter 6 reports our conclusions.

Additionally, an appendix is provided with complementary experimental results.

2 THEORETICAL FOUNDATION

In this chapter, the main theoretical concepts explored in this master's thesis are presented. We begin with the base definitions of machine learning, supervised learning, and an overview of a few algorithms that fall within this category. Afterwards, we explore data streams and how information may be extracted from them through Online Learning. Next, we present an overview of Change Detectors that allow for proper concept drift adaptation. Finally, we introduce the concepts of Active Learning and Meta-learning that provide the baseline for this work.

2.1 Machine Learning

Computer programs are composed of a set of instructions that are interpreted and executed by the computer. Traditionally, these programs are written instruction by instruction by a human programmer to execute a specific task. While this has remained the standard since the inception of the first computers, this method of solving problems is entirely dependant on the human operator's domain knowledge and abilities making it error prone and, depending on the task, extremely complex.

Machine Learning (ML) is an area of study that aims to give the computer the ability to learn and adapt to tasks without the need for a direct implementation by a human. Through the use of data relevant to the task at hand, the ML algorithm is able to learn from it, producing a model capable of executing this task with a certain level of accuracy, this accuracy being highly dependant on the quality of the data. This makes ML heavily reliant on the fields of statistics and data analysis [13].

Tasks performed by machine learning models can take on a variety of categories, following are some of the most common and studied ones [13, 14]:

- *Classification*: This concerns the problem of assigning categories or labels to items, for example, an image classification task may require that an image of an object be categorised as a dog or a cat. Classification is broadly used and usually broken into binary problems (when an item can only be assigned one of two categories) and multi-class (when an item can be assigned one of N categories). There also exist tasks where an item may receive multiple labels, this is usually referred to as multi-label classification;
- *Regression*: This concerns the prediction of a numeric value from the item. Examples of this are the prediction of a stock value in the future and the temperature of the next day. Since regression concerns real numbers, measuring the performance of

a model generally revolves around the distance from the expected value and the predicted value;

- *Clustering*: This concerns the task of assigning items to clusters based on their similarity. Unlike classification, where the categories are known, in clustering the categories are unknown and learned from the patterns found in the data itself. For example, given a large number of customers in a store, these customers can be broken into clusters based on their purchasing habits, revealing important information and in some cases uncovering unexpected patterns.

To perform these tasks, there are several techniques, broken into several categories such as Supervised Learning, Unsupervised Learning, Active Learning, etc.

2.2 Supervised Learning

Supervised Learning is a category of ML techniques that are usually composed of two critical phases: Model Training and Model Testing. The first teaches the model the task and the second tests whether the performance achieved by the model is acceptable. Figure 1 shows a common process widely followed in Supervised Machine Learning tasks that follows the procedures discussed earlier until it is finally deployed in real life situations to actually perform its task.

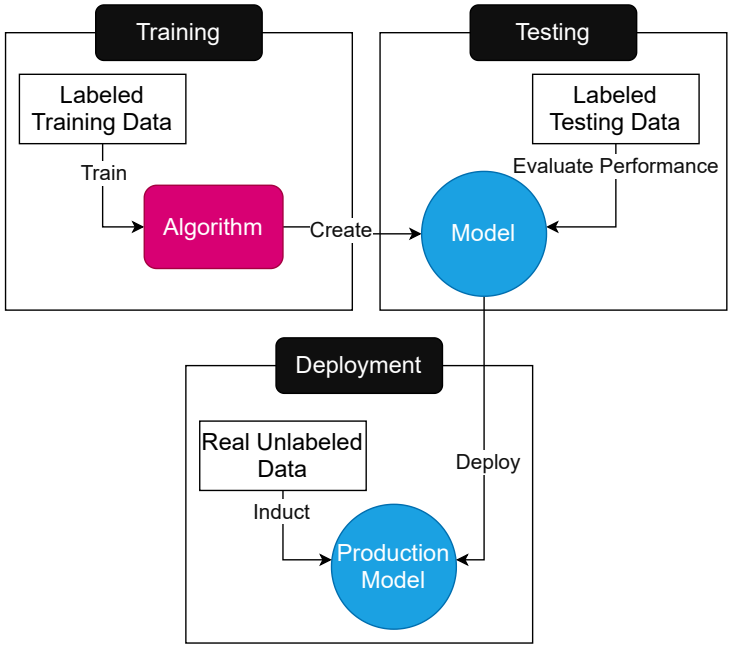


Figure 1 – General Supervised Machine Learning Procedure

This type of ML requires data composed of two elements to be trained and tested: a feature vector (represented by X) and labels (represented by y). Features are elements that describe each item and are ideally able to be used to computationally predict the

correct label for it. Features in a dataset can be enhanced by performing feature engineering or feature selection. The former is a process to create new higher quality features through the use of other features either manually or automatically [15]. The latter is the exclusion of poor quality features that may decrease the performance of the model [13]. The quality of the features impact directly the quality of the model created, a concept commonly known as Garbage in, Garbage out.

Supervised Learning is the most commonly found application in Machine Learning nowadays, but it comes with a problem that makes its deployment into real world applications difficult: the train and test phases require every item to be labeled, which usually involves domain experts, analysts, and other human agents. This process involves a cost that increases as more and more data is involved. Other categories of ML techniques try to solve, or at least lessen, the problem, such as Semi-supervised Learning, Unsupervised Learning, and Active Learning. The last of which is the one relevant to this work.

In this work, we focus on 4 supervised learning algorithms: Classification and Regression Tree (CART), Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP).

2.2.1 Classification and Regression Tree

A Decision Tree is a predictive model that aims to partition the input space into smaller hierarchical spaces through the use of nodes and splits on the features of the input data [16]. The result of this is a top-down tree where each data instance navigates the nodes through its splits until it reaches a terminal node that provides the predicted output. To navigate the nodes, each split runs a test on a specific feature (e.g., featureX > 10) and the result of this test determines the next node.

One of the most famous algorithms for decision tree creation is the Classification and Regression Tree (CART) algorithm proposed by Breiman et al.[16]. It works by inducing training data through the nodes of the tree and evaluating an impurity metric on each feature to determine the feature that should be used in the next split test. The tree grows until it reaches its maximum size (if set) or until all currently terminal nodes are pure, that is, every sample that arrives at these nodes represents a single class or value. It should be noted that while decision trees are not strictly binary, CART only generates binary trees.

As the name implies, this algorithm is capable of building decision trees for both classification and regression tasks. The difference between the two can be generalised to the choice of impurity metric, with Gini Impurity for classification and Variance for regression. Gini Impurity is detailed in Equation 2.1, where Y represents the number of

possible classes and $p(i)$ represents the probability of selecting a sample with class i.

$$G(Y) = \sum_{i=0}^{X} p(i) * p(1-i) \tag{2.1}$$

Meanwhile, for regression, CART uses the variance in the node. It is a simple measure that tells how much the values differ from one another, showing the level of impurity in the node. For both Gini and Variance, the objective is to find the split that will create two child nodes R_L and R_R from the node R, such that they present the highest amount of impurity reduction from the original R node as shown in Equation 2.2, where p represents the split being tested and I the impurity of the given node.

$$\Delta I(p, R) = I(R) - I(R_L) - I(R_R) \tag{2.2}$$

CART is very commonly used because of its low complexity and small number of hyper-parameters that need to be set, making it suitable for a large number of tasks, including time-sensitive ones. Aside from algorithm specific benefits, there are also benefits to decision trees in general, such as the intuitive interpretability provided by the tree model (that can be graphically represented) and the ability to easily calculate feature importance. Regarding disadvantages, we can list the following: the difficult interpretability of large trees; their tendency to have high variance, making them prone to overfitting; and their instability in the sense that even small changes to the dataset can result in a new completely different tree [17].

2.2.2 Random Forest

Random Forests [18] are ensemble algorithms composed of decision trees. Ensemble algorithms are composed of multiple predictors that together make up a "committee" or "ensemble". Each predictor is based on a different hypothesis on the task. These hypotheses can be built in several ways, such as feeding different subsets of input data to each predictor (known as Bootstrap Aggregating or Bagging [19]) or feeding only subsets of input features to each predictor. When data is inputted into the model, each member of the committee will then provide a vote for which output it deems correct, the output of the model is the evaluation of these votes [20].

Each tree in a random forest receives a random assortment of input features to be built. They can also be coupled with Bagging. Both of these characteristics are tunable through hyper-parameters, with the number of features fed to each tree being one of the most important, even more than the number of trees in the forest [18].

For classification, RFs evaluate the votes of each member and chooses the most popular value as the output. Some implementations also base their decisions by averaging

the prediction probabilities of each predictor instead of a single chosen value [21]. For regression, the output is chosen through the averaging of the output of each member instead.

Compared to traditional decision trees, RFs can have a lower variance thanks to their sources of randomness (random features and/or bagging) and significantly lower the chance to overfit along with providing higher tolerance to noise and outliers, providing a robust and high performing model. This makes random forests one of the most used Supervised Learning algorithms due to their high performance while also requiring few hyper-parameters. These hyper-parameters are usually restricted to the number of trees, the number of features per tree, and the tree's algorithm's hyper-parameters, which traditionally uses CART, another algorithm with a small number of hyper-parameters.

2.2.3 Support Vector Machine

Support Vector Machines (SVM) are Supervised Learning algorithms that aim to find the hyperplane that best segregates the input data according to its labels. To achieve this, the SVM algorithm tries to maximize the separation margins between the data samples found in the frontier of each class, referred as Support Vectors, through the use of a Lagrangian functional [22].

While this is sufficient for linear tasks, for non-linear problems, it is necessary to manipulate the input space through the use of kernel functions, allowing the data to be mapped to a higher dimensional space where they can be linearly separated. Many types of kernel functions may be used, some popular examples are the polynomial kernel and the radial basis function kernel.

Support Vectors may also be applied to regression problems, where it takes on the name of Support Vector Regression (SVR). In this case, the idea is to adjust the hyperplane to a loss function, originally the ϵ-insensitive loss function, a function that does not penalize errors below a set $\epsilon > 0$ [22]. As is the case with classification, non-linear tasks require a kernel to transfer the input data to a linearly separated space.

SVMs are very effective when applied in tasks with high dimensionality, even if there are more features than samples, although if the difference is too large, a proper choice and tuning of the kernel are needed to avoid overfitting [21]. The use of kernels also adds high flexibility to this method, since aside from the more common kernels, it is also possible to create custom kernels adapted to the situation.

One of the main disadvantages to SVM is the number of hyper-parameters that need to be tuned, such as the choice of kernel, their hyper-parameters (e.g. polynomial kernel's degree), and regularisation parameters, specially important in regression. Also, SVM models are not easily interpretable in the same way that a decision tree might be,

for example, since it involves complex transformations to the input space.

2.2.4 Multilayer Perceptron

A Multilayer Perceptron (MLP) is a type of Neural Network (NN), more specifically a feedforward neural network. A NN is a biologically inspired model composed of multiple neurons connected between themselves, where the output of one neuron goes through an activation function before being inputted into the other neuron. Activation functions usually work as a filter in a similar way to biological neurons. This means that data is only transmitted to another neuron if they are within a boundary. A modern example of an activation function would be the Rectified Linear Unit Function (ReLU)[23], which, as illustrated in Equation 2.3, simply truncates any negative value to 0 while leaving positive values unchanged.

$$F(x) = \max(0, x) \tag{2.3}$$

Each neuron transforms its input with the Equation 2.4, where N is the number of inputs, w_i is the weight associated with the i-th input and a_i is the i-th input, and the bias is a value set through hyper-parameters.

$$y = \sum_{i=1}^{N} w_i a_i + \text{bias} \tag{2.4}$$

The first neural network proposed was the Perceptron (or single-layer perceptron) proposed by Rosenblatt [24]. It is composed of a single neuron that receives a N sized input vector and outputs a single value. This limits Perceptrons to binary problems in simple linear tasks since a single neuron is equivalent to a linear function.

MLPs are similar to Perceptrons, but, instead of a single layer composed of a single neuron, there are multiple layers of neurons, one connected to the next, with multiple neurons in each layer. We call these layers *hidden layers* since the actual content being computed in these layers is not known ahead of time. The addition of these layers, alongside non-linear activation functions such as ReLU or, more traditionally, sigmoid, allows the MLP to solve non-linear problems that are impossible to be solved with a simple Perceptron.

The learning process of a Perceptron is simple, it calculates its output $\hat{y}$, calculates its deviance from the true value y through a loss function, and adjusts its weight values accordingly. But in MLPs it is common to use backpropagation [25], a gradient descent technique that propagates the error calculated at the output layer by layer in backwards order, adjusting the weights of each layer according to a weight optimisation solver.

MLPs have become very popular in recent times thanks to their flexibility and the rise of Deep Learning, enabled by the increasing computational power available alongside the large amount of data being generated. Deep Learning specifically is very prominent in computer vision, but NNs are able to adapt to almost any type of task. The main disadvantage of this class of models is that the more complex the task, the bigger the network, increasing the computational and data requirements exponentially.

2.3 Data Streams

A data stream is given by a potentially unbounded sequence of data that arrives in small intervals. We can define this sequence as $S =< S_1, S_2, ..., S_t, ... >$ where S_t corresponds to a data instance that arrived at time t.

The task of extracting meaningful information from data streams is called Data Stream Mining. Compared to traditional batch tasks that use static and complete datasets, Stream Mining presents many complex challenges, with the three main ones being: the *volume* of data being received, the *velocity* (or interval) of arrival and the *volatility* of the data, or changes to its patterns [26].

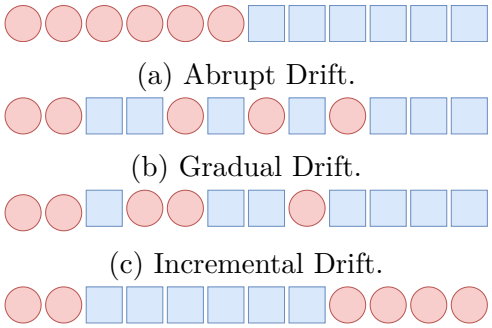
The first two challenges refer to the fact that data arrives continuously and potentially forever at very high speeds, which forces the learning tasks to be executed fast and continuously as well. Bearing this in mind, stream mining algorithms usually operate incrementally, that is, they learn from new data and discard old data that is not relevant anymore since memory is limited in real scenarios.

A common incremental learning methodology is prequential evaluation [27]. Given the current sample S_i and a model trained on $\{S_x \in S | 1 \leq x \leq i-1\}$, in this method the model predicts a value $\hat{y}_i$, where $\hat{y}_i$ may correspond to a label in a classification task or a real value in regression tasks. Following the prediction, the model is then trained on S_i. The performance of the model is updated at each new sample by comparing $\hat{y}_i$ with the true value y_i.

2.3.1 Concept Drift

Concept drifts refer to the third challenge of Stream Mining, *volatility*. It means that as new data arrives, their patterns or underlying distribution may change over time. Concept drifts may be categorised in a large variety of types, but they are not standardised. Webb et al.[28] propose a framework to formally define concept drift categories broken into multiple features that may define a drift, such as duration, recurrence, or magnitude.

For our purposes, we make use of the categories defined by Webb et al.[28], but,



(a) Abrupt Drift.

(b) Gradual Drift.

(c) Incremental Drift.

(d) Recurring Drift.

Figure 2 – Types of concept drift relevant to this work.

since the number is quite extensive, we only explore 4 types of concept drift:

- Abrupt: Occurs when change occurs in a single step. An example is when a stock market crashes and its value suddenly drops;
- Gradual: Occurs when the concept changes gradually from the old concept to the new concept that may involve small changes or large abrupt changes;
- Incremental: Similar to gradual changes, the difference is that its transition from the old to the new concept is steady in such a way that the distance from the old concept increases at the same time that the distance to the new concept decreases. In other words, the probability of samples regarding the old concept appearing steadily decreases while samples regarding the new concept steadily increase;
- Recurring: These types of drift happen when old concepts are observed again after they have been drifted away. An example would be Internet usage: during the day the usage is high and variable, while during the night it tends to drop and remain constant, the recurrence would be the day usage reappearing after the night passes.

Note that recurring is not an exclusive type of drift and may coexist with any of the others. Figure 2 shows a visualization example for each of the 4 types of drift defined earlier.

Concept drifts are phenomena that add complex challenges to the task of stream mining, since their presence may invalidate all the knowledge acquired by the model earlier and heavily deteriorate their performance.

2.3.2 Time Series vs Data Streams

Time Series and Data Streams are two concepts that while they are usually referred as separate, their similarities are often too high in a way that data streams may

even be seen as a special instance of time series [29]. A characteristic that may differentiate the two concepts is the possible lack of temporal independence observed in data streams. Temporally independent data streams in real scenarios are extremely uncommon however.

Indeed, as will be seen in future sections, all the benchmark data streams used in this work are temporally dependant, which we can classify as time series as well.

This important definition allows us to interchangeably use methods applicable to time series in data streams and vice versa, such as meta-feature extraction based on time series concepts.

2.4 Online Learning

Online and Incremental learning are terms without a universally agreed definition. Some authors define Online learning as learning from a data stream without storing data samples, unlike Incremental Learning that stores them [30]. For the purposes of this work, we use the terms interchangeably to define the process of learning from a data stream, regardless of sample storage.

Online learning often requires the use of algorithms adapted to the task so that memory and performance constraints, usually not present in traditional offline learning, are not violated.

2.4.1 Hoeffding Tree

Hoeffding Trees [1] are incremental decision trees that exploit the principle of Hoeffding Bounds to deal with infinitely large datasets where each sample can be read at most once in a short constant time. Hoeffding Bounds assure that the attribute chosen for the tree leaf split after reading n samples would be the same even if infinite samples were read by a margin of error ϵ.

To calculate this, we choose a function G (e.g., Information Gain [31]). Let a sample X be composed of a sequence of m attributes, such that $X =< x_1, x_2, \ldots, x_m >$. For each X in a sequence of n samples, we calculate G for all its attributes. Let $G(x_f)$ and $G(x_s)$ be the first and second highest values respectively among all $G(x_j)$ and that $\Delta G = G(x_f) - G(x_s)$. For a given δ, Hoeffding Bounds guarantee that x_f is the correct choice for the split with a probability of $1 - \delta$ if n examples were read at the node being trained and $\Delta G > \epsilon^2$.

The most popular implementation of Hoeffding Tree is the Very Fast Decision Tree proposed by Domingos et al.[1] that applies several optimizations to further accelerate its training process. G is usually chosen to be either Information Gain or the Gini Index.

Although very efficient for static stream distributions, Hoeffding trees have limitations regarding concept drifts and their performance tends to degrade with their presence. To overcome these limitations, adaptive versions were proposed, such as HAT-ADWIN[32] that uses an ADWIN change detector [33] to detect performance changes at the node level and build an alternate tree when drift is detected. This tree will eventually replace the main tree when its accuracy is surpassed.

2.5 Change Detection

To properly apply Online Learning techniques in data streams, dealing with changes is unavoidable. Coping with changes and concept drifts can be done either by adapting the learner at regular intervals without any knowledge of when a change happened or by first detecting the change and then triggering an adaptation in the learner. The former can be exemplified with fixed sliding windows, an interval of n samples from the stream that the learner adapts itself to. The latter is done through the use of change detectors: systems or algorithms that monitor the data stream, looking for various drift indicators [34].

Although simple in principle, the use of fixed windows can introduce a hard problem that needs to be addressed: the size of the window. A small window will allow the learner to rapidly adapt itself to changes that occur recurrently, but this can also limit the performance of the model during more stable periods. Meanwhile, a large window will allow the model to achieve higher performance during stable moments due to the larger number of samples, but it will be slow to adapt to the changes that occur.

Thus, change detectors attempt to avoid having to deal with this trade-off. As mentioned previously, change detectors look for indicators in the stream that signal changes. These indicators may be provided by some different measures, with performance attributes of the learner (such as accuracy or error) being especially common [34, 35, 36]. By finding the drift points in the stream, the detector is then able to create sliding windows with variable sizes, to better accommodate the current behavior of the stream.

2.5.1 ADWIN

ADWIN [33] (short for ADaptive WINdowing) is an algorithm used to detect changes in streams and build sliding windows around their drift points. It is very popular due to its simplicity and the use of a single hyperparameter δ.

ADWIN works by keeping a sliding window W that at each time t, a new value x_t is appended from the data stream. At each new addition, the average of two sub-windows of W are compared and if their difference is higher than ϵ_{cut}, then older data is dropped until the difference between the sub-windows becomes lower than ϵ_{cut}.

To calculate ϵ_{cut}, let us define the two sub-windows of W as W_0 and W_1, such that $W = W_0 \cdot W_1$. With this in mind, we can define the length of W as $n = n_0 + n_1$, where n_0 and n_1 are the lengths of each sub-window respectively. Equation 2.5 details the computation of ϵ_{cut}, where σ^2_W corresponds to the observed variance of the elements in W, m corresponds to the harmonic mean of the lengths of the sub-windows as seen in Equation 2.6 and $\delta' = \delta/n$, δ being the algorithm's sole hyperparameter as said beforehand.

$$\epsilon_{cut} = \sqrt{\frac{2}{m} \cdot \sigma^2_W \cdot ln(\frac{2}{\delta'})} + \frac{2}{3m} ln(\frac{2}{\delta'}) \tag{2.5}$$

$$m = \frac{1}{1/n_0 + 1/n_1} \tag{2.6}$$

ADWIN is a powerful change detector that provides good results with minimal tuning. A major downside though is the inability to process multi-dimensional data. ADWIN is univariate and receives a single signal to determine changes in the stream. To circumvent this, a common tactic is to first process the data by a learner and then input its output value (usually an error value) into the ADWIN model instead of using the raw stream data.

2.6 Active Learning

Since traditional supervised learning requires the knowledge of y for every sample being used for training, and in the context of prequential evaluation, this means every new data sample, real world applicability becomes restricted since labelling every sample is a costly and inefficient process that can not keep up with the streaming velocity.

Active Learning allows the model to learn more efficiently by only requiring a subset of the total samples. This can greatly reduce the number of labels required while also speeding up the training process since less data is used. To achieve this, ActL is broken into many strategies and scenarios. Two examples of strategies are Membership Query Synthesis and Selective Sampling [10].

Membership Query Synthesis [37] synthesises new samples based on the underlying distribution of the area of uncertainty in the input domain to train the model instead of using the already existing data. This strategy is not widely used thanks to its complexity compared to other methods along with a limitation regarding its ability to generate samples capable of being interpreted, an important attribute if the labeler is a human oracle [38].

Selective Sampling [39], on the other hand, utilises the input data samples directly. The decision of which samples are to be used for training is left up to a query strategy.

Selective Sampling is broken into two types: Pool-based and Stream-based.

The former assumes a static input domain and is therefore better performing in batch training tasks. It maintains a large pool of unlabelled samples and every iteration they are ranked based on the query strategy and the first n samples are queried. Iterations are triggered based on the situation, it may be time-based (e.g. every hour), statistic-based (e.g. when the learner reaches a certain accuracy) or any other condition [11].

On the other hand, Stream-based Selective Sampling is very straightforward. As new data samples are received, they are processed by the query strategy and it will then decide if it is worth querying this label or not and if it is within the budget if it is defined. Data samples that are not queried are simply discarded from the training process.

Regarding query strategies, there is a multitude of different algorithms, but for this work, we concern ourselves with Uncertainty Sampling, the most commonly used due to its simplicity and speed while also providing better results compared to other query strategies, for example, Random Sampling [10].

Uncertainty Sampling works by extracting a metric (or uncertainty measure) from the data sample and comparing it to a threshold (Z). If the threshold is exceeded, the sample is deemed interesting from a training perspective and is queried.

There also exist many uncertainty measures, the simplest one encompasses only binary classification tasks, where the class probability for the current data sample is extracted from the model and if their values are close to 0.5 (in this case, Z is a number close to 0.5), then the model is very uncertain and the sample should be queried. However, the most general and widely used measure is *entropy*[40] with a Z defined by hyper-parameter [10]. When we refer to Uncertainty Sampling, in this work, we also refer to Uncertainty Sampling applied with Entropy and is defined by Equation 2.7.

$$x_H^* = \underset{x}{\operatorname{argmax}} - \sum P_\theta(y_i|y_x)\log_2 P_\theta(y_i|y_x) \tag{2.7}$$

One of the biggest problems related to the use of Active Learning in streaming environments is dealing with concept drift [41] and imbalanced classes [42]. Since the structure of the data may change, the Active Learning configuration may become outdated and may even negatively impact the learning process.

2.7 Meta-learning

We can define meta-learning as the process of learning from previous experiences and applying this knowledge to a new task [43]. This can be achieved in a variety of approaches and methodologies. A popular approach is Stacked Generalisation or simply

Stacking [44], where a set of base-learners (referred as level-0 models) are inducted in a training set T_{train}, this training set being representative of the actual input data for the task. The output of these learners, along with their expected output, is then used to create a new training set T'_{train} that is then inducted into a set of meta-learners (level-1 models). These models have the responsibility to find the best way to interpret the outputs of the base-learners and generate the proper output for the task.

Although in this example only 2 layers were specified (level-0 and level-1 models), Stacking allows for multiple layers added on top of each other, hence the name. Each layer also allows for different choices of algorithms (e.g. Decision Trees on level-0 and Linear Regression on level-1), differentiating it from other common methods of ensemble learning, such as bagging or boosting. However, the reason that Stacking is considered a meta-learning approach is that the transformation of the training sets from T_{train} to T'_{train} is able to convey meta-knowledge about the base-learners [45].

Another approach is the use of a meta-learner that is trained before the task at hand. This is achieved through the creation of a meta-database composed of meta-features extracted from the actual tasks. These meta-features may be generated from the results of the model, such as accuracy and error, or statistics and information from the input data. The trained meta-model is then deployed to the tasks where it may influence it in a variety of ways, such as tuning hyper-parameters or choosing the proper classifier for the task.

A meta-feature is a measure extracted from n samples from a dataset with the objective of representing characteristics from the dataset or a section of it. Taking this into consideration, we can see that a meta-feature is composed not only of a measuring function, but also of a summarisation function as defined by Rivolli et al.[46], who define a meta-feature formally as the function f in Equation 2.8, where m is a measure function and σ is a summarisation function with h_m and h_s being hyper-parameters for m and σ respectively.

$$f(\mathcal{D}) = \sigma(m(\mathcal{D}, h_m), h_s) \tag{2.8}$$

2.7.1 Time series Meta-features

In data streaming situations, or more specifically time series tasks, specific types of meta-features may be extracted that provides information covering important aspects found in a data stream and how they correlate as time passes and new data arrives.

In Table 1 we describe a few of these possible meta-features, they are all provided by the TSFEL library [47]. These same meta-features are also all the meta-features being used in this work. Each meta-feature can be categorised by a descriptor that hints at

the type of information extracted from the time-series, they are: Energy, Correlation, Neighbourhood, Format, Distance, Statistics and Entropy.

Table 1 – Time-series meta-features. s is the signal vector, t is the time vector represented by $(i/fs)_{i=0}^{N}$ where fs is the sampling rate and N is the size of s, and Δs is the discrete derivative of s defined by $\Delta s = s_{i+1} - s_i$.

Name	Descriptor	Function
Absolute energy	Energy	$\sum_{i=0}^{N} s_i^2$
Total energy	Energy	$\frac{\sum_{i=0}^{N} s_i^2}{t_N - t_0}$
Autocorrelation	Correlation	$\sum_{n \in Z} s(n)\overline{s(n-l)}$
Centroid	Neighbourhood	$\frac{\sum_{i=0}^{N} t_i \times s_i^2}{\sum_{i=0}^{N} s_i^2}$
Neighbourhood peaks	Neighbourhood	Count the peaks found using continuous wavelet transform [48] with n (default = 0) neighbours
Peak to peak distance	Format	$\lvert max(s) - min(s) \rvert$
Negative turning points	Format	$\sum_{i=0}^{N-1} turn(\Delta s_{i+1}, \Delta s_i)$ where $turn(x,y) = \begin{cases} 1, \text{if } x < 0 \text{ and } y > 0 \\ 0, else \end{cases}$
Positive turning points	Format	$\sum_{i=0}^{N-1} turn(\Delta s_i, \Delta s_{i+1})$ where $turn(x,y) = \begin{cases} 1, \text{if } x < 0 \text{ and } y > 0 \\ 0, else \end{cases}$
Slope	Format	m coefficient from fitted equation $s = mt + b$
Zero crossing rate	Format	$\sum_{i=1}^{N} 1_{\mathbf{R}<0}(s_{i-1}si)$ where $1_{\mathbf{R}<0}$ is an indicator function [49]
Mean absolute diff	Distance	$mean(\lvert \Delta s \rvert)$
Mean diff	Distance	$mean(\Delta s)$
Median absolute diff	Distance	$median(\lvert \Delta s \rvert)$
Median diff	Distance	$median(\Delta s)$
Signal distance	Distance	$\sum_{i=0}^{N-1} \sqrt{1 + \Delta s_i^2}$
Sum absolute diff	Distance	$\sum_{i=0}^{N-1} \lvert \Delta s_i \rvert$
Area under the curve	Statistical	$\sum_{i=0}^{N} (t_i - t_{i-1}) \times \frac{s_i + s_{i-1}}{2}$
Entropy	Entropy	$-\sum_{x \in s} P(x) log_2 P(x)$

3 RELATED WORK

In this chapter, we present related work in the literature that explores the application of Active Learning and Meta-learning to Stream Mining. It also exposes the shortcomings that these previous studies found.

3.1 Active Learning applied to Stream Mining

As we presented before, ActL struggles to deal with concept drift and imbalance in classes. A few solutions are found in the literature, such as training a new classifier in the background when the old classifier begins to drop accuracy and finally replacing it with the new one when drift is detected by a drift detection algorithm. This algorithm also controls the labelling rate of instances to accommodate for new concepts [50]. This allows the model to prevent losing accuracy due to the model trying to adapt to the changes without modifying anything else.

Shan et al.[8] proposed a framework that allows for concept drift adaptation during stream classification. It uses an ensemble composed of a stable classifier that always learns the latest data and an array of dynamic classifiers that learn on late sliding windows. With the use of Active Learning, using both Uncertainty Sampling and Random Sampling, the framework is capable of maintaining high accuracy levels and gradually stabilising in the advent of concept drift.

Liu et al.[51] proposed a framework that relies on two metrics to decide whether an instance should be labeled. The first one, called local density, is a metric based on Ebbinghaus's law of human memory cognition [52] to create a sliding window based on the forgetting curve from which the local density is calculated. The second one is the classical uncertainty sampling grounded on the representativeness of the instance and the confidence that the classifier can make the right decision. This method achieves good stability and performance even in the presence of gradual concept drift while also keeping only one classifier unlike the previously seen methods. However, it also requires the setting of several hyper-parameters with many options to tune without a good optimisation method. Also, under abrupt concept drift scenarios its superiority to other methods is not obvious.

In the context of sampling, Bouguelia et al.[53] proposed a method that provided an adaptive threshold value that was adjusted according to the error rate of the model. While this method was created with the "sufficient weight" sampling method proposed, it is also usable in regular Uncertainty Sampling contexts. The disadvantage of this method is that it does not consider concept drifts.

It can be seen that propositions that address Active Learning in the context of Stream Mining tend to either aim for performance and general adaptability or concept drift handling at the cost of complexity and resource usage.

3.2 Meta-learning for Stream Mining

Meta-learning has some recent applications to data stream mining in literature. One of the preliminary proposals was made by Rossi et al.[54]. They proposed a framework for algorithm selection on stream regression tasks using meta-learning. The idea is based on extracting meta-features from the data stream bounded by sliding windows, creating meta-instances that feed the meta-model. The meta-model chooses one or more algorithms to be trained on the current sliding window of data. Different from traditional meta-learning frameworks, the models are always retrained instead of keeping static until recalled by the meta-learner. It also allows for ensembles to be formed when the meta-learner chooses more than one algorithm to build models over the current data.

Anderson et al.[55] proposed a framework to deal with recurring concept drifts during stream classification. The authors' proposal is based on maintaining a repository of classifiers alongside the current classifier being used. When drift is detected, a meta-learner will then choose a classifier from this repository, which it judges as the fittest for the new structure of the data. The framework keeps training the current and maintained classifiers using new instances from right before the drift (when the detector signals a warning state). Once a new drift is detected, one of the classifiers that were being used is stored in the repository and the low accurate discarded before the meta-learner selects a new classifier.

Drift detection was also addressed with MtL by Yu et al. [56] who propose a framework that not only detects concept drifts but also classifies them into their different types (Sudden, Gradual, Incremental, and Recurring). Another contribution is related to the capacity of reducing the cold start effects often present in traditional drift detection methods. This framework can be split into two main phases: a training phase where the meta-model is trained and the detection phase, where the meta-model is deployed. During the training phase, meta-features from examples of various types of concept drift are extracted and used to train the meta-model. At the detection phase, the meta-model is set to detect concept drift while also adapting itself dynamically by training on relevant examples selected by Stream-based Active Learning.

It is worth mentioning the concern related to drift adaptation by all meta-learning over data stream approaches presented. The concept drift phenomenon requires attention, which demands a considerable change in the trained model and even an algorithm shift after a drift. In other words, a drift adaptation is beyond model updating, requiring

hyperparameter tuning or the use of other algorithm biases.

3.3 Chapter Conclusion

Technique	Reference	Drift Adaptation	# Classifiers	Tuning Complexity	Main Contribution
ActL	Krawczyk et al.[50]	Yes	3 (1 Active, 1 Background, 1 Drift Detector)	Medium	Maintain accuracy even under drift and small budgets
	Shan et al.[8]	Yes	1 + N (Ensemble: 1 Active and N Dynamic)	Medium	Maintain accuracy by controlling labelling rate
	Liu et al.[51]	Yes (Gradual)	1	High	Maintain stability with minimal resource usage
	Bouguelia et al.[53]	No	1	Low	Adjust labelling rate according to error rate while maintaing high accuracy
MtL	Rossi et al.[54]	Yes	1 to N (when building ensembles) + 1 Meta-model	Medium	Select the most appropriate algorithm for new arriving data
	Anderson et al.[55]	Yes (Abrupt and Recurring)	3 (1 Active, 1 Drift Detector, 1 Meta-model) + N (Repository)	Low	Maintain accuracy and adaptability under recurring drift by reusing classifiers
ActL + MtL	Yu et al.[56]	Yes	1	Medium	Provide an accurate drift detector with less reliance on humans
ActL + MtL	This work	Yes	1 + 1 Meta-model + 1 Drift Detector	Low	Provide a competitive Active Learning solution with automatic Uncertainty Sampling tuning allowing for on-the-fly adaptation to drifts

Table 2 – Summary of related studies in the use of Active Learning and Meta-learning for Stream Mining compared to the solution proposed in this work. Note that every Active Learning technique always has the advantage of selective labelling, while Meta-learning techniques rely on all labels being available.

This chapter highlighted previous works on both the usage of Active Learning and Meta-learning on data streams. It became visible that, although previous studies provided interesting solutions to the problems related to concept drifts in data streams, they still possess a good amount of shortcomings. Table 2 presents a summary of these studies, alongside the approach proposed in this work.

It is notable the lack of hybrid techniques that use both Active Learning and Meta-learning. This exposes a few limitations on each work. In the Active Learning section, one study only addresses one category of drift [51] while another does not address it at all [53], focusing instead on lowering labelling efforts. The other studies [50, 8] present techniques based on multiple classifiers and labelling rate control (not entirely unlike the

one presented in this work). The problem introduced by these works is that the addition of extra classifiers introduces a considerable memory and performance overhead.

Meanwhile, in the Meta-learning section, we have studies entirely focused on drift adaptation [54, 55]. Since they lack the use of Active Learning, there is no labelling rate reduction while also keeping the problem of multiple classifiers being trained with added overhead. Notable is also the fact that the study presented by Anderson et al.[55] only addresses two categories of drift.

Finally, we have the sole use of both techniques to provide a hybrid solution by Yu et al.[56]. Its aim is entirely unrelated though, with a focus on simply detecting concept drift points and not on classifier training and adaptation.

Through our work, we aim to provide a solution that addresses a few of these shortcomings. Primarily, we look to lower the tuning complexity and resource usage (in the number of classifiers, for example) that plagues the majority of Active Learning solutions when change adaptation is attempted. On the side of Meta-learning we try to provide a new use for these techniques where instead of selecting and changing entire classifiers, we only change one hyperparameter.

In the next chapter, we introduce our approach to the use of Active Learning combined with Meta-learning and the setup used for our experiments.

4 MATERIALS AND METHODS

This chapter presents the details of our approach, the datasets used for our experiments, and the methodology used. Our approach, named Z_MtL, is compared to the use of traditional supervised machine learning alongside traditional active learning with fixed Z values.

4.1 Proposed Approach

Through the use of Stream-based Active Learning alongside Uncertainty Sampling for label querying, our stream classifier is able to be trained with a lower labelling cost. But, this introduces a hyperparameter that requires tuning for each task, the Z value or *uncertainty threshold*, which determines the amount of uncertainty or entropy necessary for the ActL model to deem a sample worthy labelling.

Our approach, referred as Z_MtL, aims to dynamically tune Z by adding a meta-model on top of the ActL model. This meta-model, grounded on Meta-learning theory following the meta-feature training approach exposed earlier, is responsible for selecting an appropriate Z for each stream chunk. Since a stream may change its behaviour through concept drift, the meta-model decides on a new Z routinely, through a set trigger, e.g., a change detector [1, 33].

We propose to employ a trigger based on ADWIN change detector [33], which detects possible changing behaviour, indicating a concept drift. When a new possible drift point is detected, a window is formed between the last drift point (or beginning of the stream) and this new drift point. The data samples contained within this window are used to extract meta-features, which feed our meta-model towards outputting a new Z for the Uncertainty Sampling. It is worth mentioning that different change detectors could perform as triggers.

Our proposal is based on meta-learning supported by temporal meta-features. These meta-features are less complex to calculate than features of other categories in the TSFEL library. Our experiments also demonstrated that the exclusion of other features has no visible performance impact. The workflow implementing our meta-learning approach is composed of five steps as Figure 3 shows. These steps are:

1. *Meta-Feature Extraction* is a step devoted to describing the event stream characteristics based on temporal lightweight time series features [47].

2. *Meta-Target Definition* identifies the best Z value that provides a suitable trade-

off between accuracy and low label querying. Since this is a data-driven step, it is necessary to discover suitable Z values able to cover the search space of possibilities under the cited trade-off assumptions.

3. *Meta-Database*, through this step, the meta-features and meta-targets are combined, forming meta-instances used to train the meta-model.

4. *Meta-Learner* is a step grounded on machine learning for inducing the meta-model using the obtained meta-instances. *Meta-Model*, is the final model able to output the recommendation of a proper Z.

5. *Meta-Recommending*, when ADWIN detects a change, its samples (stream chunk) have their features extracted towards predicting a new Z by the meta-model. The meta-model outcome is inputted into the Uncertainty Sampling to work labelling instances to the stream classifier.

4.2 Experimental Setup

Our setup was designed to answer the following research questions (RQ):

- **RQ1:** Is our Uncertainty Sampling tuning method (Z_MtL) competitive with the standard fixed values?
- **RQ2:** Does Z_MtL allow Uncertainty Sampling to support high classification accuracy when facing concept drifts?
- **RQ3:** Does Z_MtL adapt well to concept drifts by limiting queries when needed?
- **RQ4:** How well does our method compare with the regular supervised learning method (with all labels)?

To answer our research questions and demonstrate the contributions provided by our approach, we conduct experiments on different scenarios: synthetic and real-life streams affected or not by concept drifts. We select the Very Fast Decision Tree (VFDT)[1] with default hyper-parameters as our stream classifier, using the *scikit-multiflow*[57] implementation. There exist different classifiers, such as SVFDT[2] and ensembles [5]. However, the results based on the VFDT could provide with clear insights and possible limitations of our proposal.

The reason for this is that since the VFDT algorithm does not perform any drift adaptation or labelling reduction efforts, it does not generate any bias in the results of our experiments. As such, we are able to confidently assert that improvements related to the challenges our work addresses were indeed enabled by our framework.

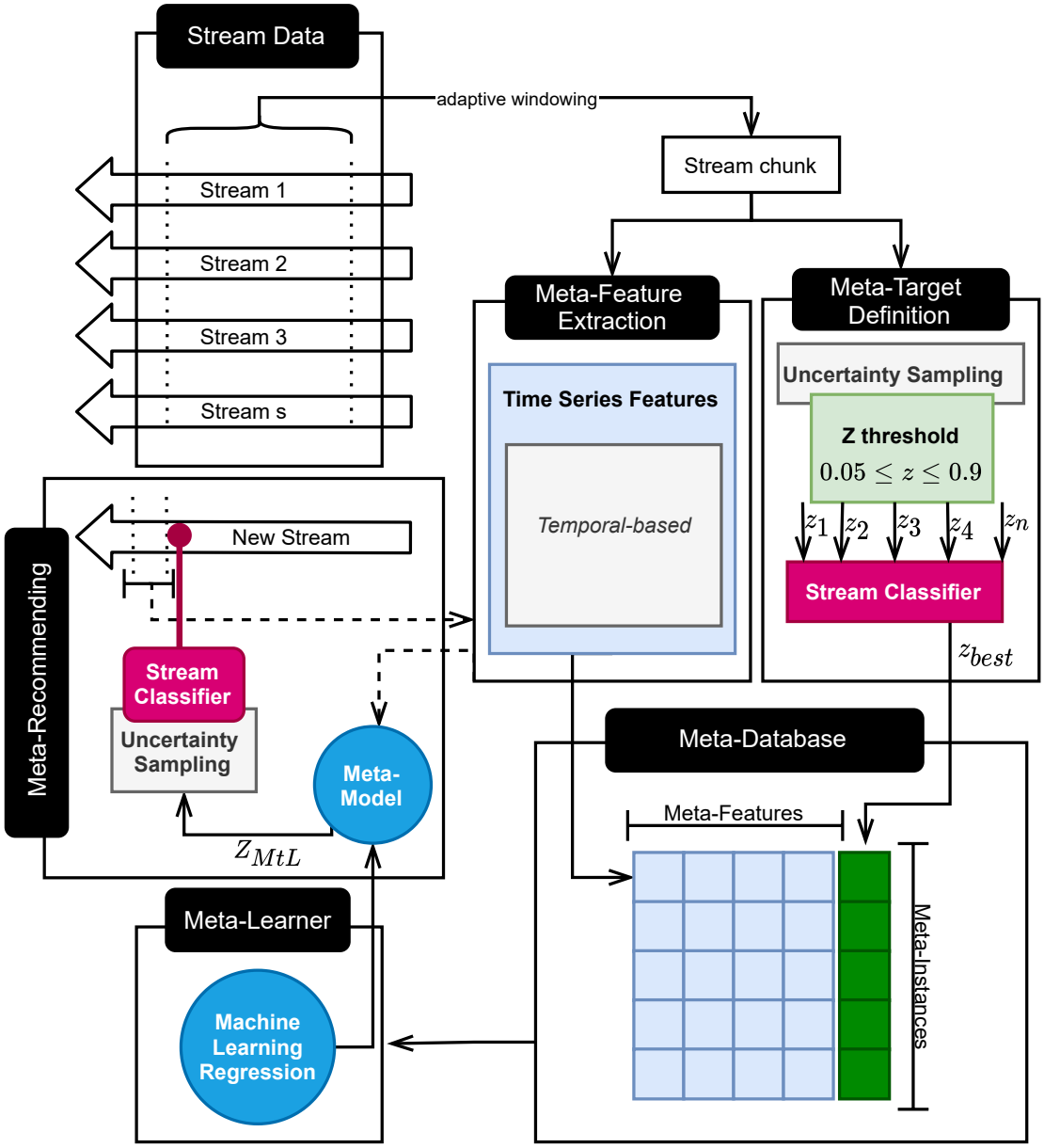


Figure 3 – Proposed approach overview

Our experimental setup can be split into four main sections: i) Meta-model training, ii) Dynamic threshold Active Learning benchmarks, iii) Fixed threshold Active Learning benchmarks, and iv) Supervised training benchmarks.

4.2.1 Meta-model Training

The meta-learning approach proposed in this paper is grounded on gathering knowledge from previous suitable Z values from diverse streams. We took advantage of several stream benchmarks and their variations to create a robust and rich source of stream data. We created a meta-database exploring a stream repository composed of 2026 stream datasets (meta-instances) generated from 7 benchmark generators [57, 58, 59, 60, 61, 62, 63]: *HyperplaneGenerator*, *LEDGeneratorDrift*, *MIXEDGenerator*, *RandomRBF-*

GeneratorDrift, *RandomTreeGenerator*, *SineGenerator*, and *STAGGERGenerator*. Each stream has approximately 300,000 samples, randomly affected by different drifts (*Gradual*, *Recurring* and *Abrupt*) covering binary and multi-class classification problems.

Note that these generated datasets are not the same ones that will be used to benchmark our framework. Instead, these streams are solely used for meta-database construction, hence their ephemeral and randomised nature.

We used ADWIN[33] to segment the stream into chunks that had their features extracted to compose the meta-instances in our meta-database. ADWIN was chosen over other algorithms thanks to its simplicity, low tuning requirements, and high performance and speed. For its one hyperparameter, δ, we applied the value of $10e^{-4}$ due to the detection performance observed in the preliminary tests. We further discuss our choice of δ in Chapter 5.

Since ADWIN is univariate, the value fed to the algorithm was the prediction result from a Hoeffding Tree making predictions on the top of the generated stream, 0 if it predicted wrongly, and 1 if predicted correctly. The ADWIN model then detects changes when the Hoeffding Tree changes its behaviour by suddenly making too many mistakes or too many hits. This same principle is applied in the *Meta-Recommending* phase, where the prediction result of the stream active learning model is fed to the ADWIN model.

Meta-features need to address the challenge of representing the stream behaviour related to a suitable Z, the threshold of Uncertainty Sampling. Each stream chunk is assessed on a particular Z by a representative set of descriptors. Moreover, it is worth noting that the meta-feature extraction step should have a low computational cost. The meta-features used are present in Table 3. Each extracted meta-feature is provided as its mean, minimum, maximum, and standard deviation values over all features or dimensions of the original stream.

In our experiments, the meta-feature extraction was performed using the TSFEL[47] library. The sampling frequency hyper-parameter was fixed to 20%. This hyper-parameter determines the amount of samples that should be sampled for each unit of time, for example: when set to 20%, it means that 1 sample is received for every 5 units of time, since $20\% = \frac{1}{5}$. Since the datasets used, both for meta-database generation and benchmarking, lack the information about their sample rate and most meta-features that require this value to be set are not being used (in our case only 3 features were used), we chose this value empirically.

To choose the algorithm of our meta-model, we evaluated 4 regression algorithms: Support Vector Machine (linear and polynomial kernel), CART Decision Trees, Multi-Layer Perceptron (with several architectures), and Random Forest.

Table 3 – Time-series meta-features used in this work. Symbols for the function definitions are the same as Table 1.

Name	Descriptor	Function
Absolute energy	Energy	$\sum_{i=0}^{N} s_i^2$
Total energy	Energy	$\frac{\sum_{i=0}^{N} s_i^2}{t_N - t_0}$
Autocorrelation	Correlation	$\sum_{n \in Z} s(n)\overline{s(n-l)}$
Centroid	Neighbourhood	$\frac{\sum_{i=0}^{N} t_i \times s_i^2}{\sum_{i=0}^{N} s_i^2}$
Neighbourhood peaks	Neighbourhood	Count the peaks found using continuous wavelet transform [48] with n (default = 0) neighbours
Peak to peak distance	Format	$\lvert max(s) - min(s) \rvert$
Negative turning points	Format	$\sum_{i=0}^{N-1} turn(\Delta s_{i+1}, \Delta s_i)$ where $turn(x,y) = \begin{cases} 1, \text{if } x < 0 \text{ and } y > 0 \\ 0, else \end{cases}$
Positive turning points	Format	$\sum_{i=0}^{N-1} turn(\Delta s_i, \Delta s_{i+1})$ where $turn(x,y) = \begin{cases} 1, \text{if } x < 0 \text{ and } y > 0 \\ 0, else \end{cases}$
Slope	Format	m coefficient from fitted equation $s = mt + b$
Zero crossing rate	Format	$\sum_{i=1}^{N} 1_{\mathbf{R}<0}(s_{i-1} si)$ where $1_{\mathbf{R}<0}$ is an indicator function [49]
Mean absolute diff	Distance	$mean(\lvert \Delta s \rvert)$
Mean diff	Distance	$mean(\Delta s)$
Median absolute diff	Distance	$median(\lvert \Delta s \rvert)$
Median diff	Distance	$median(\Delta s)$
Signal distance	Distance	$\sum_{i=0}^{N-1} \sqrt{1 + \Delta s_i^2}$
Sum absolute diff	Distance	$\sum_{i=0}^{N-1} \lvert \Delta s_i \rvert$
Area under the curve	Statistical	$\sum_{i=0}^{N} (t_i - t_{i-1}) \times \frac{s_i + s_{i-1}}{2}$
Entropy	Entropy	$-\sum_{x \in s} P(x) log_2 P(x)$

The meta-model needs to predict, as a regression task, the Z value that provides the maximum accuracy with the lowest possible number of queries. Values of Z that meet this requirement were found using the following algorithm for each window added to the meta-database used for training the meta-model:

1. Run a regular active learning classification task with a set of different possible Z values and store their mean accuracy and number of queries made;
2. For a set selection margin S, select all Z values that provided accuracy values within the range of $[MaxAcc - S, MaxAcc]$, where $MaxAcc$ is the maximum accuracy obtained from all Z values tested;
3. Sort the list of selected Z values based on the number of queries made;
4. Select the first element in the sorted list, this is the chosen target Z value.

4.2.2 Benchmark Datasets

For our benchmarks, we used 34 benchmark datasets from multiple sources. They are listed in Table 4, the number column is also used with reference to them in the other

tables.

They are essentially split into 8 groups: CTU-13 [60], Electricity [58], RandomRBF [59], LED24 [16], Hyperplane [63], Poker Hand [64], SEA [62] and Insects [61].

The CTU-13 based datasets contain botnet network traffic mixed with normal and background traffic captured in the University Czech Republic, in 2011. Each dataset corresponds to a different scenario containing a specific malware and each row of data represents a request. The challenge is to determine whether each request corresponds to normal background or botnet traffic.

The Electricity dataset was described by Harries and Wales [58] and contains data collected from the electricity market in New South Wales, Australia. Every five minutes the price changes based on supply and demand that may be influenced by other elements such as weather and season. As such, prices are not stationary and may be changed abruptly. In our experiments, we used a normalised version of this dataset.

The RandomRBF datasets are generated by creating a random set of centers for each class with a weight, a central point per attribute, and a standard deviation each. New instances are generated by choosing a random center by considering the weight of each, with the attributes being randomly offset from the chosen center and its class being the center itself [59].

The LED24 datasets are based on the example given by Breiman et al.[16] where 7 LEDs representing a segment in a LCD are set on or off with a small percentage of a faulty led, the task is to recognise which digit it represents. Aside from the original 7 led values, 17 extra segments, useless in digit representation, are added to make the problem harder. A percentage of noise is further added on top of the generated data [59].

The Hyperplane dataset is built by defining a rotating hyperplane, defined by Equation 4.1 for a d dimensional space where x_i is the i-th coordinate of x, and each instance is a random point in said space labelled 1 if $\sum_{i=1}^{d} w_i x_i \geq w_0$ and 0 if $\sum_{i=1}^{d} w_i x_i < w_0$[63]. Change is introduced by adding drift to each weight feature with $w_i = w_i + d\sigma$ where σ is the probability that the direction of change is reversed and d is the change applied to each example [57]. Our version of the dataset used 10 features with 2 of them being subjected to drift, our σ is set to 0.1 with a magnitude of change set to 0.0, 0.05% of noise is also added.

$$\sum_{i=1}^{d} w_i x_i = w_0 \tag{4.1}$$

The Poker Hand dataset obtained from the UCI Machine Learning repository [64] is based on the dataset described by Cattral et al.[65] that contained more classes

and posed a more difficult problem. Each instance in the dataset represents a hand of 5 cards (2 attributes per card, one determining the rank and the other the suit) from a standard 52 card deck. Each class represents one of 10 Poker hands, including the lack of a recognised poker hand. This dataset aims to be easy to analyse, but hard to find solutions thanks to the nature of the rules of the game [65].

The SEA dataset is a dataset capable of demonstrating abrupt changes, it is composed of 3 attributes with values ranging from 0 to 10 randomly, where only the first 2 are relevant. The class is chosen by running the first 2 features through a function selected out of 4 other functions, the drift is obtained by randomly changing the function during instance generation. The functions are defined by a condition set by $(att1 + att2 \leq \theta)$, where if the condition is satisfied, then the class is 1, else it is 0. The functions differ by their setting of θ which are 8, 9, 7, or 9.5 [62].

The Insects datasets are real-world datasets collected from the wing-beat frequency of different species of mosquitoes with varying temperatures. The datasets contain 33 features, each related to the energy sum of frequency peaks and harmonics positions [61]. The datasets are split into 6 categories:

- **Incremental:** Over the stream, the temperature is incrementally increased from 20 to 40°C;
- **Abrupt:** Five sudden changes are observed through the stream, the first instances are collected at the temperature of 30°C and then abruptly change to 20°C, after a while it changes to 35°C, and then three similar abrupt changes happen until the end of the stream;
- **Incremental-gradual:** The stream begins with instances at around 37°C and incrementally decreases to 35°C, then it begins a period of gradual change where the temperatures intercalate between 35 and 23°C until it rests at 23°C. At the end of the stream, the temperature increases incrementally to 27°C.
- **Incremental-abrupt-reoccurring:** Three recurring cycles of incremental changes in temperature occur in increases from 20 to 40°C, also, there is an abrupt change at the end and the beginning of each cycle;
- **Incremental-reoccurring:** Three cycles of incremental change also occur here, the first is an incremental increase of temperature from 20 to 40°C, the second is a decrease from 40 to 20°C ,and, finally, the third increases to 40°C.
- **Out-of-control:** There is a lack of patterns in this category, which contains data collected in uniformly random order since each instance is sampled uniformly at each time, this dataset is theoretically drift-free.

Each of these categories, except for out-of-control, makes up two datasets, one with a balanced class distribution and another imbalanced [61].

Number	Name	Instances	Features	Classes	Drifts	Type
1	CTU-13 - Scenario 1 [60]	2,824,636	11	2	38	Real
2	CTU-13 - Scenario 2 [60]	1,808,122	11	2	35	Real
3	CTU-13 - Scenario 3 [60]	4,710,638	11	2	5	Real
4	CTU-13 - Scenario 4 [60]	1,121,076	11	2	3	Real
5	CTU-13 - Scenario 5 [60]	129,832	11	2	1	Real
6	CTU-13 - Scenario 6 [60]	558,919	11	2	9	Real
7	CTU-13 - Scenario 7 [60]	114,077	11	2	3	Real
8	CTU-13 - Scenario 8 [60]	2,954,230	11	2	3	Real
9	CTU-13 - Scenario 9 [60]	2,087,508	11	2	43	Real
10	CTU-13 - Scenario 10 [60]	1,309,791	11	2	15	Real
11	CTU-13 - Scenario 11 [60]	107,251	11	2	19	Real
12	CTU-13 - Scenario 12 [60]	325,471	11	2	29	Real
13	CTU-13 - Scenario 13 [60]	1,925,149	11	2	39	Real
14	Electricity [58]	45,312	7	2	27	Real
15	RandomRBF 250k samples, 50 features [59]	250,000	50	2	9	Synthetic
16	RandomRBF 500k samples, 10 features [59]	500,000	10	2	20	Synthetic
17	RandomRBF 1M samples, 10 features [59]	1,000,000	10	2	25	Synthetic
18	LED24 1M samples, 0% noise [59]	1,000,000	24	10	10	Synthetic
19	LED24 1M samples, 10% noise [59]	1,000,000	24	10	1	Synthetic
20	LED24 1M samples, 20% noise [59]	1,000,000	24	10	0	Synthetic
21	Hyperplane [63]	250,000	10	2	4	Synthetic
22	Poker Hand [64]	829,201	10	10	153	Synthetic
23	SEA [62]	60,000	3	2	4	Synthetic
24	Insects - Abrupt Imbalanced [61]	452,044	33	6	73	Real
25	Insects - Incremental Imbalanced [61]	143,323	33	6	43	Real
26	Insects - Incremental-gradual Imbalanced [61]	355,275	33	6	61	Real
27	Insects - Incremental-reoccurring Imbalanced [61]	452,044	33	6	91	Real
28	Insects - Incremental-abrupt Imbalanced [61]	452,044	33	6	84	Real
29	Insects - Abrupt Balanced [61]	57,018	33	6	12	Real
30	Insects - Incremental Balanced [61]	24,150	33	6	14	Real
31	Insects - Incremental-gradual Balanced [61]	52,848	33	6	33	Real
32	Insects - Incremental-reoccurring Balanced [61]	79,986	33	6	47	Real
33	Insects - Incremental-abrupt Balanced [61]	79986	33	6	56	Real
34	Insects - Out-of-control [61]	905,145	33	24	21	Real

Table 4 – Benchmark stream datasets used in our experiments

These datasets are used in all of the following benchmark experiments.

4.2.3 Benchmark Experiments

Our experiments compared three frameworks around the benchmark datasets: Z_MtL, Fixed Threshold Active Learning, and traditional supervised learning with all labels. Our methodology of performance evaluation was prequential evaluation.

The experiments concerning our approach were set up as follows: at the start of the evaluation, Z was set to 0.5, as soon as the ADWIN model detects a new window, meta-features are then extracted from the last window and fed to the meta-model, the meta-model will then output the new Z, substituting the previous value. As this concerns our approach, these experiments are used to answer all Research Questions.

The fixed thresholds for the second method were defined as: 0.05, 0.1, 0.2, 0.5, and 0.7. Each of these thresholds was used once for each dataset, resulting in 204 threshold and dataset combinations. These experiments are used to answer RQ1.

Finally, the supervised learning experiments are straightforward, following a prequential evaluation, each new sample is tested on the model for performance evaluation and then immediately used to train it. We use the comparisons from these experiments to answer RQ4.

5 RESULTS

This chapter shows the results obtained from the experimental setups defined earlier. First, we compare the performance of the meta-model training with a few different classifiers. Afterwards, we analyse the benchmarking results from each framework and compare them.

5.1 Meta-model Training

As determined in the preceding chapter, to choose our meta-model algorithm, we evaluated the performance of 4 different algorithms. To test these regressors, we ran 10 repetitions of 10-fold cross-validation strategy for each algorithm and compared their RMSE.

The results from these experiments can be seen in Table 5. By analyzing these results, we can conclude that Random Forest was the most capable of these options with a RMSE of only 0.241 ($\pm$0.006). Our meta-model was built without any tuning procedure, since after testing we did not observe any relevant improvements over the default hyper-parameters.

Table 5 – Performance for each meta-model candidate across 10 repetitions of 10 cross-validation runs.

Name	RMSE	
	Mean	STD
CART	0.284	0.020
MLP (2 Layers: 25x25, L-BFGS Solver)	0.780	0.122
Random Forest	0.241	0.006
SVM (Linear)	0.305	0.017
SVM (Polynomial)	0.594	0.085

With our choice made, an importance analysis was performed under each meta-feature with the aid of our chosen meta-model algorithm, the Random Forest. This allowed us to observe that the Energy features are the most important with a score of 0.055, followed by Correlation (0.016) and Neighbourhood (0.010). Entropy meta-features have the lowest importance with a score of 0.001 as Figure 4 shows.

In addition to the meta-model algorithm, we also need to choose a δ value for our ADWIN model. We evaluated three options: $10e^{-4}$, $10e^{-3}$ and $10e^{-2}$. For each value of δ we created a new meta-database, which was used to train a meta-model. Finally, this

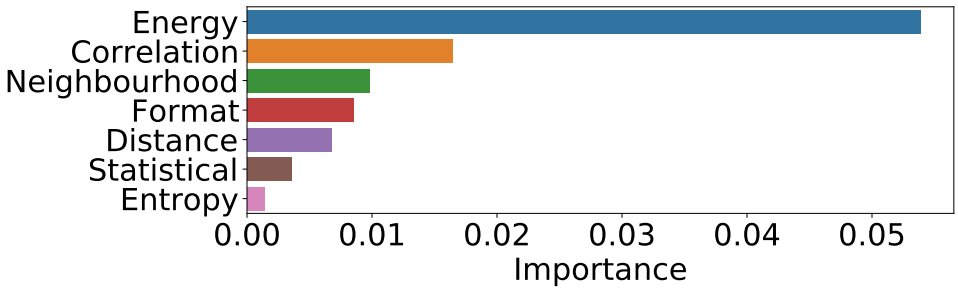


Figure 4 – Temporal-based meta-feature importance values for the Meta-model.

meta-model is induced on all the benchmark datasets. We then decided on the δ value based on their performance in accuracy and query percentage.

Table 6 demonstrates our results, represented by the mean accuracy across all datasets and the mean number of queries per dataset. Although the differences are marginal, we can see that the value $10e^{-4}$ has higher accuracy than the other values, while also keeping the second lowest query percentage. This motivated us to choose this as the value of δ in our experiments.

Table 6 – Comparison between different δ values for ADWIN.

δ	Mean Acc	Mean Queries
0.0001	89.62%	54.79%
0.001	89.44%	54.28%
0.01	89.35%	56.34%

5.2 Benchmark Experiments

As we specified in Chapter 4, we proposed a series of Research Questions that we aim to answer with our experiments, helping us better understand the impact of this work. Table 7 presents the Z value (including our method, Z_MtL) that best performed in each dataset according to their accuracy, alongside other important data: the actual accuracy value, the distance from Z_MtL, and the distance from the use of All Labels.

Table 8 aggregates the statistics values for each method and how they compare to each other across all datasets. This allows us to develop a bigger picture of each method discussed in this work. For further consultation, Table 11 in the Appendix aggregates the data regarding accuracy for all methods segregated by dataset.

5.2.1 Research Question 1

As specified earlier, we ran fixed threshold active learning intending to answer RQ1. It is possible to observe several similar high accuracy results across all static config-

Table 7 – The best Z value (including Z_MtL) for each dataset according to their accuracy, every value is in percentages. Z_MtL Δ illustrates the difference in mean accuracy between the Best Z and Z_MtL (Best Z − Z_MtL Δ). Likewise, All Labels Δ illustrates the difference in mean accuracy between the Best Z and the use of All Labels, marked in bold are the values where the Best Z had higher accuracy than All Labels (Best Z − All Labels Δ).

N	Dataset	Best Z	Mean Acc	Z_MtL Δ	All Labels Δ
1	CTU-13 - Scenario 1	Tie (All)	99.26	-	-0.58
2	CTU-13 - Scenario 2	Tie (All)	98.88	-	-0.7
3	CTU-13 - Scenario 3	Tie (All)	99.81	-	-0.19
4	CTU-13 - Scenario 4	Tie (All)	99.92	-	-0.07
5	CTU-13 - Scenario 5	Tie (All)	99.70	-	-0.16
6	CTU-13 - Scenario 6	0.05	99.31	0.03	-0.64
7	CTU-13 - Scenario 7	Tie (All)	99.92	-	-0.03
8	CTU-13 - Scenario 8	Tie (0.1, 0.2, 0.5, 0.7, Z_MtL)	99.83	-	-0.15
9	CTU-13 - Scenario 9	Tie (All)	96.89	-	-1.64
10	CTU-13 - Scenario 10	Tie (All)	94.76	-	-5.2
11	CTU-13 - Scenario 11	Tie (All)	98.88	-	-0.57
12	CTU-13 - Scenario 12	Tie (0.1, 0.2, 0.5, 0.7, Z_MtL)	99.25	-	-0.19
13	CTU-13 - Scenario 13	Tie (All)	97.57	-	-1.89
14	Electricity	0.05	79.86	1.85	**+1.13**
15	RandomRBF 250k samples, 50 features	0.2	97.74	0.64	-0.7
16	RandomRBF 500k samples, 10 features	0.05	87.09	1.42	-2.8
17	RandomRBF 1M samples, 10 features	0.05	89.07	1.7	-2.06
18	LED24 1M samples, 0% noise	Tie (All)	99.82	-	-0.17
19	LED24 1M samples, 10% noise	0.1	73.12	0.48	-0.61
20	LED24 1M samples, 20% noise	0.2	51.07	0.03	-0.05
21	Hyperplane	0.05	87.73	0.96	-1.43
22	Poker Hand	0.2	50.55	7.77	-23.54
23	SEA	0.05	84.83	0.22	**+0.11**
24	Insects - Abrupt Imbalanced	0.05	61.31	0.72	-4.84
25	Insects - Incremental Imbalanced	0.7	63.35	1.28	-3.73
26	Insects - Incremental-gradual Imbalanced	0.1	65.86	1.12	**+8.19**
27	Insects - Incremental-reoccurring Imbalanced	0.2	63.98	2.36	**+1.02**
28	Insects - Incremental-abrupt Imbalanced	0.1	62.34	2.15	-2.25
29	Insects - Abrupt Balanced	Z_MtL	58.05	-	**+4.27**
30	Insects - Incremental Balanced	0.1	49.30	0.26	-2.92
31	Insects - Incremental-gradual Balanced	Z_MtL	75.13	-	**+14.24**
32	Insects - Incremental-reoccurring Balanced	Z_MtL	56.26	-	**+2.95**
33	Insects - Incremental-abrupt Balanced	Z_MtL	59.61	-	**+1.95**
34	Insects - Out-of-control	0.05	50.44	0.98	-5.85

Table 8 – Aggregate accuracy values for each method containing Minimum, Mean, Median, Maximum and Standard Deviation. The best value among the methods (excluding All Labels) is marked in bold. Underlined values in the All Labels row indicate that it lost to an Active Learning method.

Method	Min	Mean	Median	Max	STD
0.05	41.98	80.19	**87.41**	**99.92**	19.86
0.1	43.18	80.21	87.02	**99.92**	19.62
0.2	**49.21**	**80.46**	86.68	**99.92**	**19.33**
0.5	45.85	79.76	86.61	**99.92**	19.92
0.7	33.67	78.46	83.75	**99.92**	20.71
Z_MtL	42.78	80.19	86.22	**99.92**	19.67
All Labels	51.12	80.87	84.72	100.00	18.47

urations and the dynamic one (Z_MtL), particularly on all 13 datasets from the CTU-13 group where ties happened between all values of Z. This shows that the different uncertainty sampling setup could not lead to improvements in these datasets, since all experimented Z deliver similar predictive performance. On the one hand, static Z values

of 0.05, 0.1, 0.2 led to highly accurate results. On the other hand, 0.7 and 0.5 delivered lower accuracy with very few instances where they represented the best value among all datasets.

It is important to observe that the stream datasets affected by concept drift (datasets from 24 to 33) revealed different results of the best accurate Z value. Instead of the most restrictive ones taking over the top rankings, we observed the presence of medium and high Z, e.g., 0.2 and 0.7, achieving better performance than 0.05 and 0.1. Furthermore, there are particular real and synthetic datasets in which Z_MtL obtained the best performance (datasets 29, 31, 32, and 33), even boosting the accuracy of using all labels. This is visible in Table 7 where the values in the All Labels Δ column are positive (marked in bold). A great part of them underwent the concept drift effect.

Observing the winning Z values in Table 7, we can also see that less restrictive values, such as 0.05 and 0.2 dominate the table with their highest accuracy values of (99.31%) and (97.74%) respectively, losing only to the non usage of Active Learning. Although not shown in this table, the Z value of 0.1 keeps a competitive value in most cases, right behind 0.05 but never outright surpassing it.

Further evidence of this can be seen in Table 8, where we can see that 0.05, 0.1, and 0.2 kept the highest median values across all methods, outperforming even the use of All Labels. The same is true for the mean values, although in this case All Labels is superior.

The more restrictive the Z, the more reduced the accuracy is since it demands less labels, with a Z of 0.7 leading to the lowest average performance (78.46%), also interesting to note that this Z achieved the highest standard deviation, showing that it was the least stable among the other values.

Table 9 gives us a comparison between the fixed Z values and Z_MtL regarding the trade-off between accuracy and querying rate. This value was obtained by simply dividing the accuracy by querying rate, meaning that higher values indicate higher accuracy with fewer samples and vice versa. We can observe that Z_MtL presented the best trade-off among 6 datasets: datasets 19, 22, 25, 26, 28, and 33. Although it loses to 0.7 in some datasets, they remain very competitive while the other Z values lag in almost all instances.

With these insights, we can conclude that Z_MtL is indeed competitive with fixed thresholds, since its performance does not deviate too far from the less restrictive Z values and even surpasses them in a few cases, answering RQ1.

Table 9 – Trade-off between Accuracy and Querying rate for each dataset, higher is better. The bold values indicate the Z value that produced the best trade-off.

N	Datasets	Fixed Z					Z_MtL
		0.05	0.1	0.2	0.5	0.7	
1	CTU-13 - Scenario 1	**1.09**	**1.09**	**1.09**	**1.09**	**1.09**	**1.09**
2	CTU-13 - Scenario 2	**1.09**	**1.09**	**1.09**	**1.09**	**1.09**	**1.09**
3	CTU-13 - Scenario 3	**1.24**	**1.24**	**1.24**	**1.24**	**1.24**	**1.24**
4	CTU-13 - Scenario 4	**1.43**	**1.43**	**1.43**	**1.43**	**1.43**	**1.43**
5	CTU-13 - Scenario 5	**1.09**	**1.09**	**1.09**	**1.09**	**1.09**	**1.09**
6	CTU-13 - Scenario 6	1.07	**1.10**	**1.10**	**1.10**	**1.10**	**1.10**
7	CTU-13 - Scenario 7	**1.10**	**1.10**	**1.10**	**1.10**	**1.10**	**1.10**
8	CTU-13 - Scenario 8	**1.38**	**1.38**	**1.38**	**1.38**	**1.38**	**1.38**
9	CTU-13 - Scenario 9	**1.16**	**1.16**	**1.16**	**1.16**	**1.16**	**1.16**
10	CTU-13 - Scenario 10	**1.19**	**1.19**	**1.19**	**1.19**	**1.19**	**1.19**
11	CTU-13 - Scenario 11	**1.17**	**1.17**	**1.17**	**1.17**	**1.17**	**1.17**
12	CTU-13 - Scenario 12	**1.42**	**1.42**	**1.42**	**1.42**	**1.42**	**1.42**
13	CTU-13 - Scenario 13	**1.20**	**1.20**	**1.20**	**1.20**	**1.20**	**1.20**
14	Electricity	1.05	1.62	2.28	45.98	**172.69**	3.66
15	RandomRBF 250k samples, 50 features	13.88	19.37	16.88	**24.72**	22.31	21.67
16	RandomRBF 500k samples, 10 features	1.53	1.66	2.11	3.53	**5.01**	4.19
17	RandomRBF 1M samples, 10 features	1.75	1.86	2.53	4.84	**6.64**	5.72
18	LED24 1M samples, 0% noise	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
19	LED24 1M samples, 10% noise	0.8	0.77	0.81	1.19	1.23	**1.25**
20	LED24 1M samples, 20% noise	0.51	0.51	0.51	**0.52**	**0.52**	0.51
21	Hyperplane	0.96	0.99	1.11	1.54	**1.89**	1.61
22	Poker Hand	6.44	7.43	3.6	12.26	6.08	**19.36**
23	SEA	0.89	0.91	1.0	2.21	**2.95**	2.35
24	Insects - Abrupt Imbalanced	0.86	0.92	0.9	1.39	**2.01**	1.58
25	Insects - Incremental Imbalanced	0.79	0.8	0.79	0.97	1.18	**1.35**
26	Insects - Incremental-gradual Imbalanced	1.16	1.26	1.36	2.07	2.09	**3.25**
27	Insects - Incremental-reoccurring Imbalanced	0.77	0.88	0.84	0.92	**1.23**	1.09
28	Insects - Incremental-abrupt Imbalanced	0.75	0.8	0.77	1.15	1.16	**1.23**
29	Insects - Abrupt Balanced	1.3	1.36	1.63	2.1	**2.78**	2.3
30	Insects - Incremental Balanced	0.86	0.96	1.0	1.29	**1.56**	1.34
31	Insects - Incremental-gradual Balanced	2.33	2.78	3.29	4.49	**6.38**	6.32
32	Insects - Incremental-reoccurring Balanced	1.13	1.22	1.61	2.22	**3.01**	2.77
33	Insects - Incremental-abrupt Balanced	1.12	1.3	1.57	2.15	2.98	**3.60**
34	Insects - Out-of-control	0.59	0.65	0.63	0.7	**1.00**	0.73

5.2.2 Research Question 2

Regarding our second research question, we notice that Z_MtL achieved a median accuracy value (86.22%) higher than the one obtained with the usage of all labels (84.72%) when using a VFDT, showing an ability to adapt to concept drift when compared to traditional supervised learning, which can have its performance deteriorated.

Focusing on finding a superior method to setup Z, we evaluated the results based on statistical analysis grounded on the non-parametric Friedman test to determine any significant differences among the fixed Z values and the dynamic recommended ones (Z_MtL) using 34 datasets. We used the post hoc Nemenyi test to infer which differences are statistically significant. As Figure 5 shows, differences between populations are significant. Particularly, we assume that there are no significant differences within Z of 0.1, 0.05, 0.2, 0.5, and Z_MtL; Z of 0.2, 0.5, and Z_MtL. All other differences are significant. In other words, Z_MtL achieved results statistically equal to the usage of the most reduced Z values, those that query more frequently for labels and provide high-accuracy values.

Moreover, it is important to note that datasets 29, 31, 32 and 33, which contain

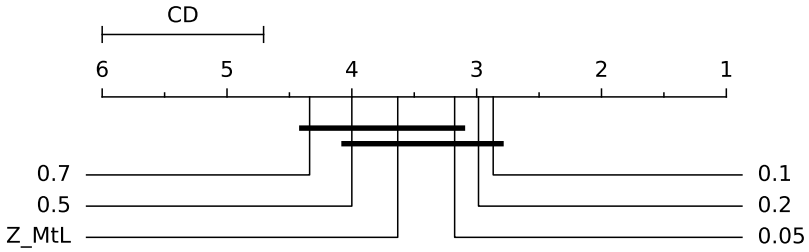


Figure 5 – Nemenyi post hoc test (significance of $\alpha = 0.05$ and critical distance (CD) of 1.29) considering the accuracy obtained using static thresholds Z (0.05, 0.1, 0.2, 0.5 and 0.7) and our proposal (Z_MtL).

concept drifts, presented the highest accuracy under Z_MtL. This along with the fact that Z_MtL is statistically equal to the less restrictive Z values allows us to answer RQ2.

5.2.3 Research Question 3

Figure 6 shows the query percentage of the evaluated datasets across the static thresholds and Z_MtL. Low Z values (0.05, 0.1, and 0.2) required an average of 70.15% of instance labels. On the other hand, the more restrictive Z (0.5 and 0.7) and the dynamic one (Z_MtL) restricted the demand for labels to around 54.78% of the total of samples. Similarly to the accuracy evaluation, it was possible to observe several similar querying numbers when classifying CTU-13 streams (datasets 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 13), as Table 10 shows. Moreover, the accuracies were similar to the usage of all labels but with fewer labels used. For example, CTU-13 - Scenario 12 achieved 99.25% of accuracy querying 69.74% of labels.

The datasets 15 (RandomRBF 250k) and 22 (Poker Hand) had a querying percentage inferior to 10% of all available instances, without compromising the accuracy when compared to the usage of all labels. Particularly, it was required almost all labels (99.95%) when processing Dataset 18 (LED24 1M samples with 0% noise).

Z_MtL recommended values were able to reach the smaller number of queries in 6 datasets (19, 22, 25, 26, 28, and 33), the most of them were affected by concept drift.

We evaluate the performance of the reduction in terms of queries requested based on a statistical analysis grounded on the non-parametric Friedman test. This allow us to determine any significant differences between the fixed Z values and the dynamic recommended ones (Z_MtL) using 34 datasets. Again, we made use of of the post hoc Nemenyi test to infer which differences are statistically significant. Figure 7 shows the Nemenyi post hoc test on the number of queries, where we can observe that differences between populations are significant. Particularly, we assume that there are no significant differences within Z of 0.7 and Z_MtL; Z of 0.7 and 0.5; Z of 0.5 and 0.2; Z of 0.2, 0.1,

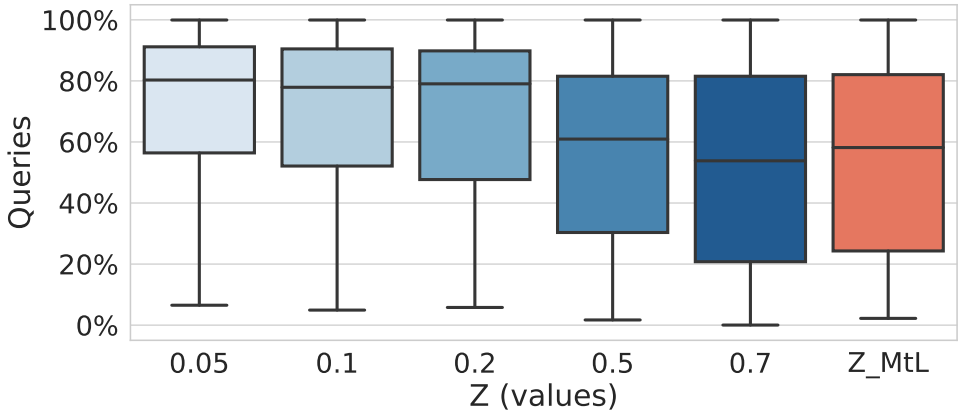


Figure 6 – Query demand all datasets for static thresholds (0.05, 0.1, 0.2, 0.5, 0.7) and our proposal (Z_MtL).

Table 10 – The best Z value (including Z_MtL) for each dataset according to their query percentage. Z_MtL Δ illustrates the difference in querying between the Best Z and Z_MtL (Best Z − Z_MtL Δ).

N	Dataset	Best Z	Query Percentage	Z_MtL Δ
1	CTU-13 - Scenario 1	Tie (All)	91.17	-
2	CTU-13 - Scenario 2	Tie (All)	90.48	-
3	CTU-13 - Scenario 3	Tie (All)	80.30	-
4	CTU-13 - Scenario 4	Tie (All)	69.99	-
5	CTU-13 - Scenario 5	Tie (All)	91.32	-
6	CTU-13 - Scenario 6	Tie (0.2, 0.5, 0.7, Z_MtL)	89.86	-
7	CTU-13 - Scenario 7	Tie (All)	90.97	-
8	CTU-13 - Scenario 8	Tie (0.1, 0.2, 0.5, 0.7, Z_MtL)	72.51	-
9	CTU-13 - Scenario 9	Tie (All)	83.63	-
10	CTU-13 - Scenario 10	Tie (All)	79.96	-
11	CTU-13 - Scenario 11	Tie (All)	84.62	-
12	CTU-13 - Scenario 12	Tie (0.1, 0.2, 0.5, 0.7, Z_MtL)	69.74	-
13	CTU-13 - Scenario 13	Tie (All)	81.53	-
14	Electricity	0.7	0.45	-20.87
15	RandomRBF 250k samples, 50 features	0.7	3.33	-1.15
16	RandomRBF 500k samples, 10 features	0.7	16.83	-3.61
17	RandomRBF 1M samples, 10 features	0.7	12.92	-2.35
18	LED24 1M samples, 0% noise	Tie (All)	99.95	-
19	LED24 1M samples, 10% noise	Z_MtL	58.17	-
20	LED24 1M samples, 20% noise	0.7	98.36	-1.01
21	Hyperplane	0.7	46.08	-7.92
22	Poker Hand	Z_MtL	2.21	-
23	SEA	0.7	28.16	-7.85
24	Insects - Abrupt Imbalanced	0.7	28.76	-9.68
25	Insects - Incremental Imbalanced	Z_MtL	45.91	-
26	Insects - Incremental-gradual Imbalanced	Z_MtL	19.94	-
27	Insects - Incremental-reoccurring Imbalanced	0.7	47.68	-8.88
28	Insects - Incremental-abrupt Imbalanced	Z_MtL	48.87	-
29	Insects - Abrupt Balanced	0.7	20.77	-4.52
30	Insects - Incremental Balanced	0.7	30.87	-5.71
31	Insects - Incremental-gradual Balanced	0.7	11.30	-0.58
32	Insects - Incremental-reoccurring Balanced	0.7	17.66	-2.66
33	Insects - Incremental-abrupt Balanced	Z_MtL	16.56	-
34	Insects - Out-of-control	0.7	49.10	-18.47

and 0.05. All other differences are significant. In other words, Z_MtL achieved results statistically equal to the usage of the highest Z value, the most restricted when querying labels.

It is important to emphasise that the dynamic recommended Z (Z_MtL) is sta-

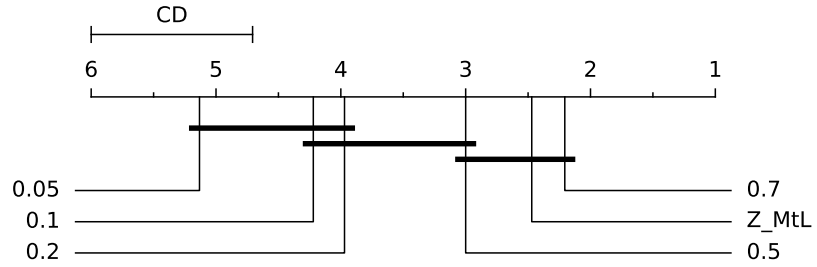


Figure 7 – Nemenyi post hoc test (significance of $\alpha = 0.05$ and critical distance (CD) of 1.29) considering the number of queries requested using static thresholds (0.05, 0.1, 0.2, 0.5 and 0.7) and our proposal (Z_MtL)

tistically similar to the less restrictive Zs (0.5, 0.1, and 0.2) in terms of accuracy. Moreover, Z_MtL values were statistically similar to the most restrictive Zs (0.7 and 0.5) when reducing the query demand. In other words, our proposed approach provides the best trade-off between accuracy and query reduction by dynamic tuning Z values using lightweight meta-features.

Additionally, we observed the accuracy of datasets 29, 31, 32, and 33 (Table 7) were boosted by the usage of Z_MtL in comparison to using all available labels to incrementally induce the VFDT. Precisely, Z_MtL improved the accuracy by 4.27% for Dataset 29 (Insects - Abrupt Balanced); 14.24% for Dataset 31 (Insects - Incremental-gradual Balanced); 2.95% for Dataset 32 (Insects - Incremental-reoccurring Balanced); and finally, 1.95% for Dataset 33 (Insects - Incremental-abrupt Balanced). All these datasets shared the presence of concept drift in a balanced classification problem.

Regarding concept drifts, Uncertainty Sampling reveals the potential of selecting the most informative features when adapting the classifier to a drift. We select four different data streams to support the discussion on drift adaptation, the importance of using dynamic Z values, and provide evidence to answer RQ3.

Figure 8 shows the results for the Dataset 25 (Insects - Incremental Imbalanced). In this image we can see how Z tends to change in accordance to concept drifts when running on Z_MtL usually by decreasing it and allowing the learner to query more instances and adapt to changes faster while increasing it in stagnated periods to avoid unnecessary querying. Since our ADWIN model is fed on the error values for each sample, drift points are usually indicative of performance degradation or stagnation within the model. This allows our framework to step in and provide a more adequate Z value.

With this, we can answer RQ3 since we can see that Z_MtL will control the querying rate in drift situations.

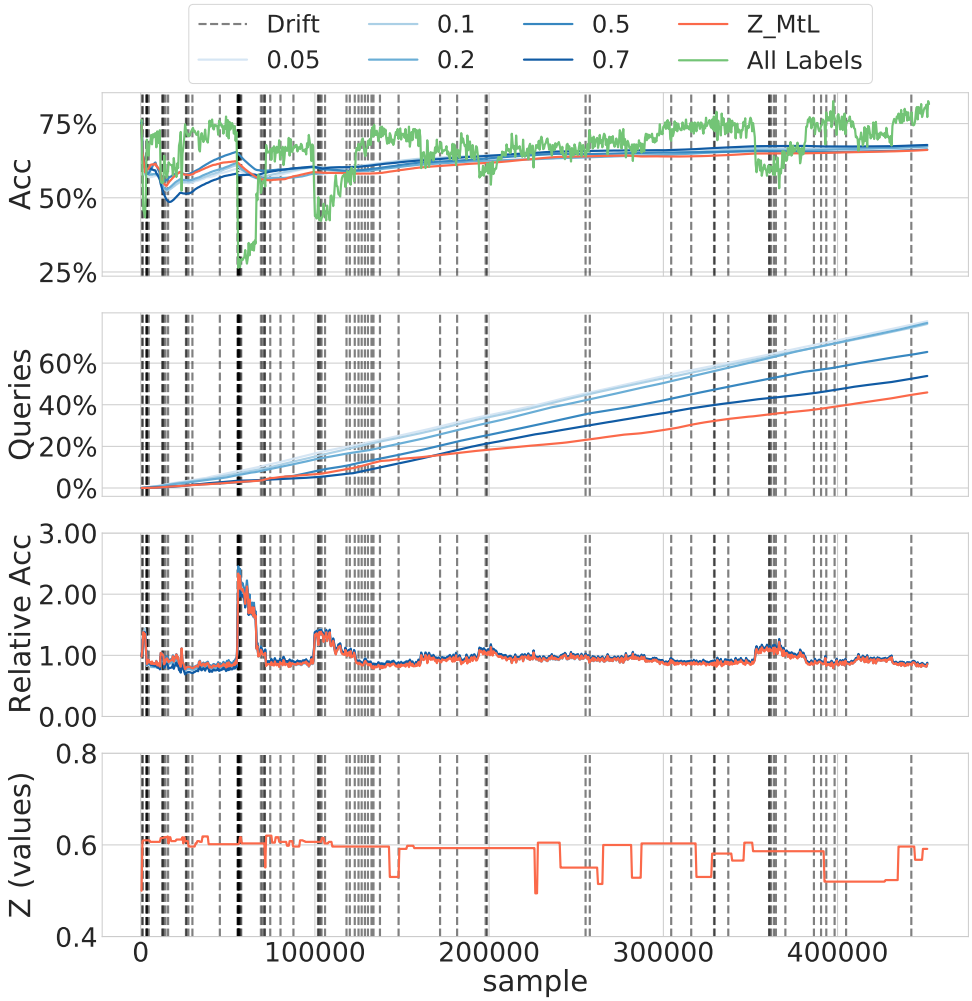


Figure 8 – Accuracy, Queries, Relative Accuracy and dynamically adjusted Z values from experiments using Insects - Incremental Imbalanced stream (Dataset 25).

5.2.4 Research Question 4

Using the Electricity stream from different perspectives as Figure 9 shows, we can observe the drift starting points and the particular changes in terms of accuracy, number of queries, and uncertainty sampling Z values. Relative accuracy in this figure refers to the accuracy for each Z value normalised to the accuracy found through the user of all labels.

Observing the accuracy, it is evident that for all labels this metric changes considerably after a drift point. On the other hand, using uncertainty sampling, a smooth and regular accuracy was obtained. Regarding the number of queries, it was not observed a strict relation between drift points and a rough modification. We can affirm that Z equals 0.05 linearly grows up throughout the stream. This behavior is similar to 0.1 and 0.2, but with a small gradient. Z_MtL started to present this behavior at 28,000, but was not devoted to a drift point. The relative accuracy was not affected. Regarding the dynamically adjusted Z (bottom chart on Figure 9), we can observe variations from 0.43

to 0.71, emphasizing the demand of tuning the threshold of Uncertainty Sampling.

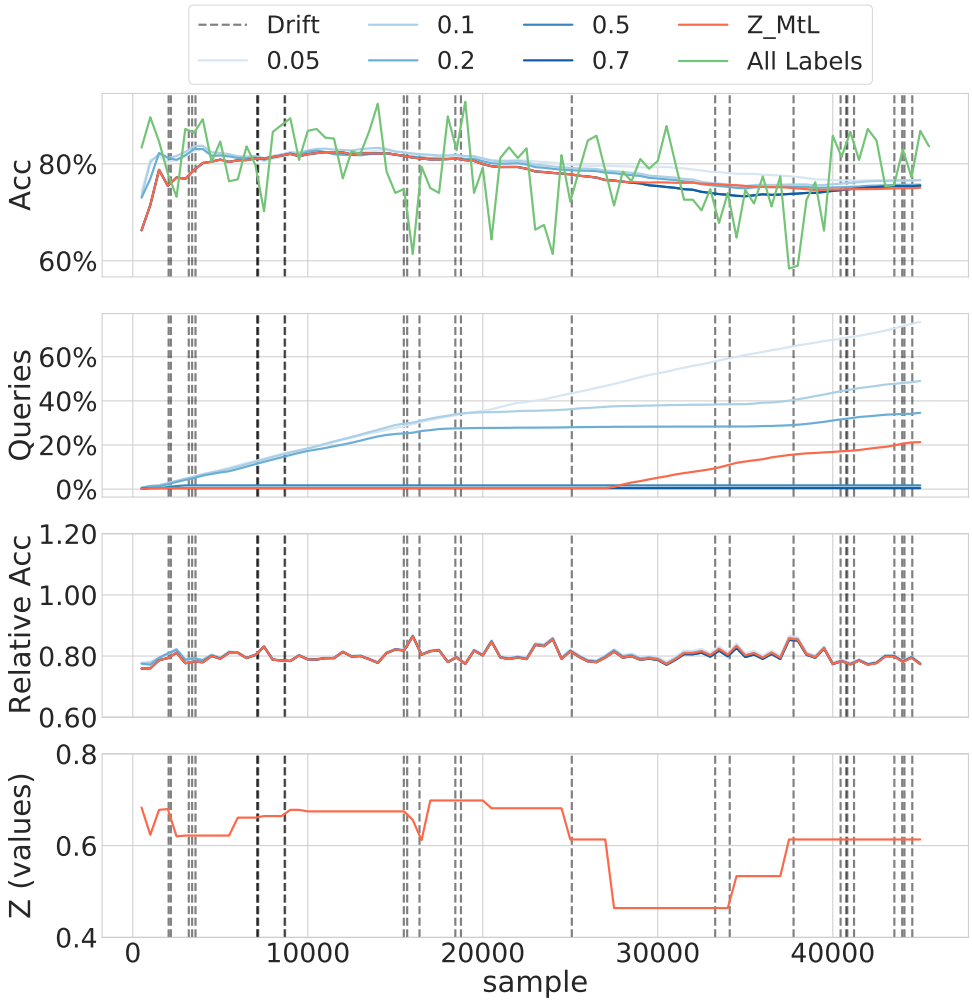


Figure 9 – Accuracy, Queries, Relative Accuracy, and dynamically adjusted Z values from experiments using Electricity stream (Dataset 14).

Observing Z_MtL results, which overcame the usage of all labels, we selected Dataset 31 (Insects - Incremental and Gradual) to explore some particular behaviors that are shown in Figure 10. Queries and Z values followed similar results. Slightly linear queries have their slope related to Z, achieving higher inclination with less restrictive values. This data stream has a gradual concept drift started at around 14,000 that reduced the accuracy of VFDT when using all labels. When using Active Learning for training, the VFDT was hardly affected by the gradual drift, presenting a smooth reduction in terms of accuracy. It is important to mention that after the concept drift, the relative accuracy surpasses 6.00, with Z_MtL delivering the most accurate result. VFDT was able to recover the performance, i.e., adjust to the drift, approximately at the point 18,000. This window from 14,000 to 18,000 was suitably handled by uncertainty sampling, leading the VFDT to achieve superior results in comparison to the usage of all labels.

The Dataset 26 (Insects - Incremental and Gradual Imbalanced) shown in Figure 11 further reinforces our observations. In this dataset, it is notable that the VFDT

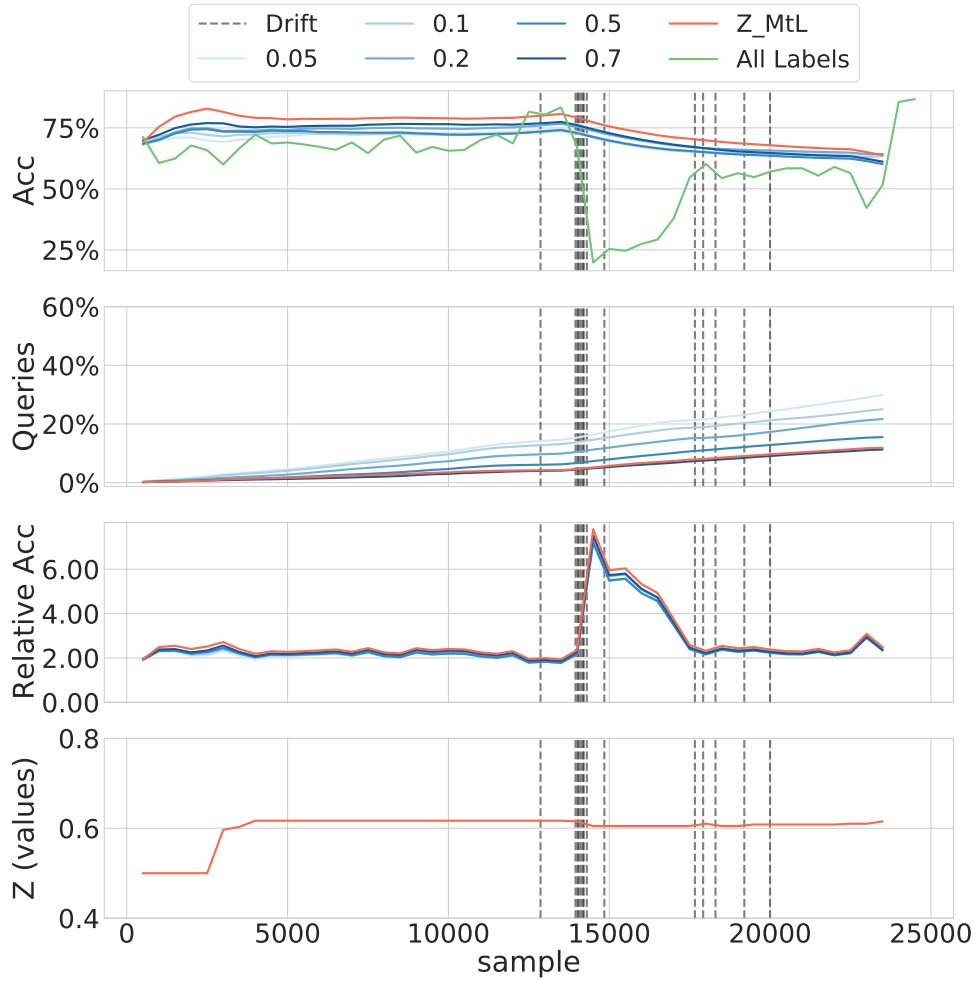


Figure 10 – Accuracy, Queries, Relative Accuracy and dynamically adjusted Z values from experiments using Insects - Incremental and Gradual Balanced stream (Dataset 31).

accuracy suffers dramatically across different points of drift when using all labels while Z_MtL remains relatively stable even if its accuracy suffers marginal losses, while also maintaining a very low query rate compared to other Active Learning techniques, never surpassing the 20% mark.

These achievements help to answer the RQ4 since scenarios over concept drift can take advantage of frequent and accurate updating of Z values. We can affirm that by selecting important features and avoiding irrelevant ones, it is possible to improve the predictive performance.

The ability to robustly adjust to concept drift, and outperforming the usage of all available labels by consequence, leads to a increased VFDT drift adaptability. This improvement was observed in most experiments with datasets affected by concept drift (datasets 25, 26, 30, 31, 32, 33, 34), half of them with dynamic tuning (Z_MtL) as the most effective approach. Furthermore, using the dynamic tuning of Z, the label query process was quite reduced, similarly to the most restrictive values.

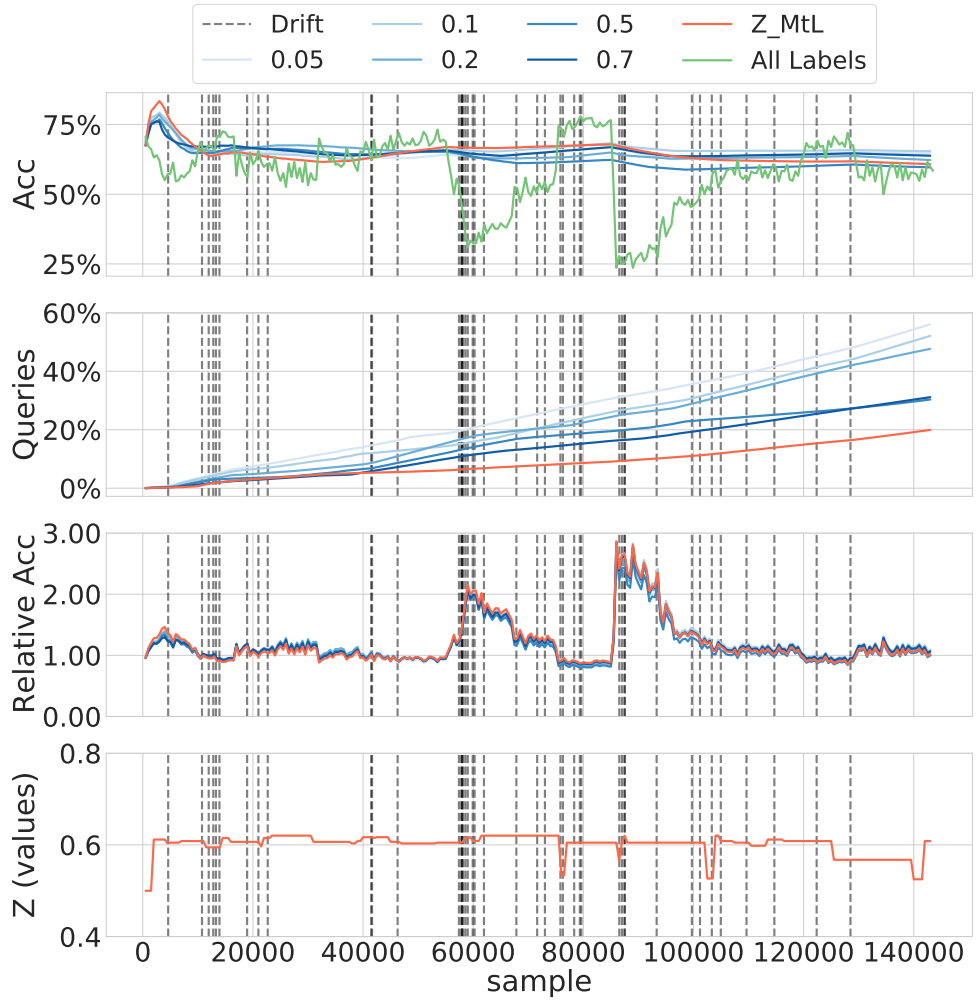


Figure 11 – Accuracy, Queries, Relative Accuracy and dynamically adjusted Z values from experiments using Insects - Incremental and Gradual Imbalanced stream (Dataset 26).

The main advantages of the proposed method are: i) Online selection of suitable Z values based on lightweight features. ii) Competitive results using Z_MtL, which can outperform the usage of all labels with fewer samples. iii) Capacity to provide concept drift adaptation for simple algorithms such as VDFT. Finally, we consider that all research questions were addressed positively, which indicates that our proposal is a promising Stream-based Active Learning taking advantage of Uncertainty Sampling for label querying.

6 CONCLUSION

Active Learning is a field that greatly enhances Stream Mining techniques by reducing the cost of labelling while keeping competitive accuracy. In contrast to a batch based approach, stream mining costs can reach very high levels due to the nature of infinite fast-arriving data. In other words, Active Learning helps us build a bridge between real-life stream mining processing and supervised approaches.

However, the use of Uncertainty Sampling requires a level of tuning that changes according to the task at hand, not only that, but concept drift can also require adaptation. Our proposal aims to reduce the overhead and costs of choosing the best Uncertainty threshold (Z) automatically through Meta-Learning. Furthermore, allowing changes and automatic tuning through the processing of the stream.

Our tests showed a significant improvement in the usage of Uncertainty Sampling when selecting samples for stream classification. By using our solution to dynamically select Z, it was possible to improve the accuracy reducing the demand of labels considerably all while keeping a steady adaptation to the introduced concept drifts.

In particular, due to our framework adapting well to the newly introduced concept drifts, we managed to obtain superior performance when compared to traditional supervised classification using all labels.

The obtained results encourage us to pursue further work in investigating the dynamic tuning of alternative Active Learning approaches, Ensemble and Adaptive Classifiers, and the use of budgeting for further querying reduction. Furthermore, our analysis regarding the trade-off between accuracy and cost encourage us to pursue better optimisation strategies regarding Z value choices for different situations.

BIBLIOGRAPHY

[1] DOMINGOS, P.; HULTEN, G. Mining high-speed data streams. In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.: s.n.], 2000. p. 71–80.

[2] COSTA, V. G. T. da et al. Strict very fast decision tree: a memory conservative algorithm for data stream mining. *Pattern Recognition Letters*, Elsevier, v. 116, p. 22–28, 2018.

[3] MANAPRAGADA, C.; WEBB, G. I.; SALEHI, M. Extremely fast decision tree. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. [S.l.: s.n.], 2018. p. 1953–1962.

[4] DIN, S. U. et al. Online reliable semi-supervised learning on evolving data streams. *Information Sciences*, Elsevier, v. 525, p. 153–171, 2020.

[5] KRAWCZYK, B. et al. Ensemble learning for data stream analysis: A survey. *Information Fusion*, Elsevier, v. 37, p. 132–156, 2017.

[6] KRAWCZYK, B.; CANO, A. Adaptive ensemble active learning for drifting data stream mining. In: *International Joint Conference on Artificial Intelligence*. [S.l.: s.n.], 2019. p. 2763–2771.

[7] LUGHOFER, E. On-line active learning: A new paradigm to improve practical useability of data stream modeling methods. *Information Sciences*, Elsevier, v. 415, p. 356–376, 2017.

[8] SHAN, J. et al. Online active learning ensemble framework for drifted data streams. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, v. 30, n. 2, p. 486–498, 2018.

[9] MARTINS, V. E.; COSTA, V. G. T. da; JUNIOR, S. B. Active learning embedded in incremental decision trees. In: *Brazilian Conference on Intelligent Systems*. [S.l.: s.n.], 2020. p. 367–381.

[10] SETTLES, B. *Active Learning Literature Survey*. [S.l.], 2009.

[11] LEWIS, D. D.; GALE, W. A. A sequential algorithm for training text classifiers. In: *International ACM-SIGIR Conference on Research and Development in Information Retrieval*. [S.l.: s.n.], 1994. p. 3–12.

[12] LU, J. et al. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 31, n. 12, p. 2346–2363, 2018.

[13] MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. *Foundations of Machine Learning, second edition*. MIT Press, 2018. (Adaptive Computation and Machine Learning series). ISBN 9780262351362. Disponível em: <https://books.google.com.br/books?id=dWB9DwAAQBAJ>.

[14] ALZUBI, J.; NAYYAR, A.; KUMAR, A. Machine learning from theory to algorithms: an overview. In: IOP PUBLISHING. *Journal of physics: conference series*. [S.l.], 2018. v. 1142, n. 1, p. 012012.

[15] NARGESIAN, F. et al. Learning feature engineering for classification. In: *Ijcai*. [S.l.: s.n.], 2017. p. 2529–2535.

[16] BREIMAN, L. et al. *Classification and regression trees*. [S.l.]: Routledge, 2017.

[17] MOLNAR, C. *Interpretable Machine Learning*: A guide for making black box models explainable. [S.l.: s.n.], 2019.

[18] BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001.

[19] BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996.

[20] DIETTERICH, T. G. et al. Ensemble learning. *The handbook of brain theory and neural networks*, MIT press Cambridge, Massachusetts, v. 2, n. 1, p. 110–125, 2002.

[21] PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

[22] VAPNIK, V. *The nature of statistical learning theory*. [S.l.]: Springer science & business media, 1999.

[23] NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: *Icml*. [S.l.: s.n.], 2010.

[24] ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958.

[25] RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986.

[26] KREMPL, G. et al. Open challenges for data stream mining research. *ACM SIGKDD explorations newsletter*, ACM New York, NY, USA, v. 16, n. 1, p. 1–10, 2014.

[27] DAWID, A. P. Present position and potential developments: Some personal views statistical theory the prequential approach. *Journal of the Royal Statistical Society: Series A (General)*, Wiley Online Library, v. 147, n. 2, p. 278–290, 1984.

[28] WEBB, G. I. et al. Characterizing concept drift. *Data Mining and Knowledge Discovery*, Springer, v. 30, n. 4, p. 964–994, 2016.

[29] READ, J. et al. Data streams are time series: Challenging assumptions. In: SPRINGER. *Brazilian Conference on Intelligent Systems*. [S.l.], 2020. p. 529–543.

[30] SAFFARI, A. et al. On-line random forests. In: IEEE. *2009 ieee 12th international conference on computer vision workshops, iccv workshops*. [S.l.], 2009. p. 1393–1400.

[31] QUINLAN, J. R. Induction of decision trees. *Machine learning*, Springer, v. 1, n. 1, p. 81–106, 1986.

[32] BIFET, A.; GAVALDA, R. Adaptive learning from evolving data streams. In: SPRINGER. *International Symposium on Intelligent Data Analysis*. [S.l.], 2009. p. 249–260.

[33] 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.

[34] GAMA, J. et al. Learning with drift detection. In: SPRINGER. *Brazilian symposium on artificial intelligence*. [S.l.], 2004. p. 286–295.

[35] KLINKENBERG, R.; RENZ, I.; AG, D.-b. Adaptive information filtering: Learning in the presence of concept drifts. 03 1999.

[36] BAENA-GARCIA, M. et al. Early drift detection method. In: *Fourth international workshop on knowledge discovery from data streams*. [S.l.: s.n.], 2006. v. 6, p. 77–86.

[37] ANGLUIN, D. Queries and concept learning. *Machine learning*, Springer, v. 2, n. 4, p. 319–342, 1988.

[38] LANG, K.; BAUM, E. Query learning can work poorly when a human oracle is used. ieee intl. In: *Joint Conference on Neural Networks*. [S.l.: s.n.], 1992.

[39] ATLAS, L. E.; COHN, D. A.; LADNER, R. E. Training connectionist networks with queries and selective sampling. In: *Advances in neural information processing systems*. [S.l.: s.n.], 1990. p. 566–573.

[40] SHANNON, C. E. A mathematical theory of communication. *Bell system technical journal*, Wiley Online Library, v. 27, n. 3, p. 379–423, 1948.

[41] ŽLIOBAITĖ, I. et al. Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, v. 25, n. 1, p. 27–39, 2013.

[42] KORYCKI, Ł.; CANO, A.; KRAWCZYK, B. Active learning with abstaining classifiers for imbalanced drifting data streams. In: *IEEE International Conference on Big Data (Big Data)*. [S.l.: s.n.], 2019. p. 2334–2343.

[43] VANSCHOREN, J. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

[44] WOLPERT, D. H. Stacked generalization. *Neural networks*, Elsevier, v. 5, n. 2, p. 241–259, 1992.

[45] VILALTA, R.; DRISSI, Y. A perspective view and survey of meta-learning. *Artificial intelligence review*, Springer, v. 18, n. 2, p. 77–95, 2002.

[46] RIVOLLI, A. et al. Characterizing classification datasets: a study of meta-features for meta-learning. *arXiv preprint arXiv:1808.10406*, 2018.

[47] BARANDAS, M. et al. TSFEL: Time Series Feature Extraction Library. *SoftwareX*, v. 11, p. 100456, 2020.

[48] DU, P.; KIBBE, W. A.; LIN, S. M. Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching. *Bioinformatics*, v. 22, n. 17, p. 2059–2065, 2006.

[49] CHEN, C.-h. *Signal processing handbook*. [S.l.]: CRC Press, 1988. v. 51.

[50] KRAWCZYK, B.; PFAHRINGER, B.; WOŹNIAK, M. Combining active learning with concept drift detection for data stream mining. In: *IEEE International Conference on Big Data (Big Data)*. [S.l.: s.n.], 2018. p. 2239–2244.

[51] LIU, S. et al. Online active learning for drifting data streams. *IEEE Transactions on Neural Networks and Learning Systems*, In Press, p. 1–15, 2021.

[52] EBBINGHAUS, H. Memory: A contribution to experimental psychology. *Annals of neurosciences*, SAGE Publications, v. 20, n. 4, p. 155, 2013.

[53] BOUGUELIA, M.-R.; BELAÏD, Y.; BELAÏD, A. An adaptive streaming active learning strategy based on instance weighting. *Pattern Recognition Letters*, Elsevier, v. 70, p. 38–44, 2016.

[54] ROSSI, A. L. D. et al. Metastream: A meta-learning based method for periodic algorithm selection in time-changing data. *Neurocomputing*, v. 127, p. 52–64, 2014.

[55] ANDERSON, R. et al. Recurring concept meta-learning for evolving data streams. *Expert Systems with Applications*, v. 138, p. 112832, 2019.

[56] YU, H. et al. Automatic learning to detect concept drift. *arXiv preprint arXiv:2105.01419*, 2021.

[57] MONTIEL, J. et al. Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, v. 19, n. 72, p. 1–5, 2018.

[58] HARRIES, M.; WALES, N. S. Splice-2 comparative evaluation: Electricity pricing. *Citeseer*, 1999.

[59] HALL, M. et al. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, ACM New York, NY, USA, v. 11, n. 1, p. 10–18, 2009.

[60] GARCIA, S. et al. An empirical comparison of botnet detection methods. *computers & security*, Elsevier, v. 45, p. 100–123, 2014.

[61] SOUZA, V. M. et al. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, Springer, v. 34, n. 6, p. 1805–1858, 2020.

[62] STREET, W. N.; KIM, Y. A streaming ensemble algorithm (SEA) for large-scale classification. In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.: s.n.], 2001. p. 377–382.

[63] HULTEN, G.; SPENCER, L.; DOMINGOS, P. Mining time-changing data streams. In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.: s.n.], 2001. p. 97–106.

[64] DUA, D.; GRAFF, C. *UCI Machine Learning Repository*. 2017. Disponível em: <http://archive.ics.uci.edu/ml>.

[65] CATTRAL, R.; OPPACHER, F.; DEUGO, D. Evolutionary data mining with automatic rule generalization. *Recent Advances in Computers, Computing and Communications*, v. 1, n. 1, p. 296–300, 2002.

Appendix

APPENDIX A – COMPLETE PERFORMANCE TABLES

The Tables 11 and 12 contain performance data for each method evaluated in our experiments for each dataset for further examination. These tables do not contain the dataset names, only their identifying numbers which can be cross-referenced in Table 4. This was done to save space in the case of Table 11 and to maintain consistency with the former table in case of Table 12.

Table 11 – Accuracy percentage of all Z values across all 34 datasets and their standard deviation. The best performance per dataset is marked in bold and the standard deviation is shown in parenthesis (also in percentages). The underlined values in the All Labels column indicates that it lost in accuracy to Active Learning techniques.

Datasets	Fixed Z					Z_MtL	All Labels
	0.05	0.1	0.2	0.5	0.7		
1	**99.26 (0.62)**	**99.26 (0.62)**	**99.26 (0.62)**	**99.26 (0.62)**	**99.26 (0.62)**	**99.26 (0.62)**	99.84 (0.81)
2	**98.88 (0.60)**	**98.88 (0.60)**	**98.88 (0.60)**	**98.88 (0.60)**	**98.88 (0.60)**	**98.88 (0.60)**	99.58 (0.71)
3	**99.81 (0.19)**	**99.81 (0.19)**	**99.81 (0.19)**	**99.81 (0.19)**	**99.81 (0.19)**	**99.81 (0.19)**	100.00 (0.04)
4	**99.92 (0.07)**	**99.92 (0.07)**	**99.92 (0.07)**	**99.92 (0.07)**	**99.92 (0.07)**	**99.92 (0.07)**	99.99 (0.07)
5	**99.70 (0.28)**	**99.70 (0.28)**	**99.70 (0.28)**	**99.70 (0.28)**	**99.70 (0.28)**	**99.70 (0.28)**	99.86 (0.27)
6	**99.31 (0.22)**	99.29 (0.22)	99.28 (0.21)	99.28 (0.21)	99.28 (0.21)	99.28 (0.21)	99.95 (0.27)
7	**99.92 (0.04)**	**99.92 (0.04)**	**99.92 (0.04)**	**99.92 (0.04)**	**99.92 (0.04)**	**99.92 (0.04)**	99.95 (0.25)
8	99.82 (0.03)	**99.83 (0.03)**	**99.83 (0.03)**	**99.83 (0.03)**	**99.83 (0.03)**	**99.83 (0.03)**	99.98 (0.08)
9	**96.89 (3.40)**	**96.89 (3.40)**	**96.89 (3.40)**	**96.89 (3.40)**	**96.89 (3.40)**	**96.89 (3.40)**	98.53 (1.74)
10	**94.76 (4.40)**	**94.76 (4.40)**	**94.76 (4.40)**	**94.76 (4.40)**	**94.76 (4.40)**	**94.76 (4.40)**	99.96 (0.18)
11	**98.88 (2.52)**	**98.88 (2.52)**	**98.88 (2.52)**	**98.88 (2.52)**	**98.88 (2.52)**	**98.88 (2.52)**	99.45 (2.97)
12	99.22 (0.39)	**99.25 (0.38)**	**99.25 (0.38)**	**99.25 (0.38)**	**99.25 (0.38)**	**99.25 (0.38)**	99.44 (1.52)
13	**97.57 (0.45)**	**97.57 (0.45)**	**97.57 (0.45)**	**97.57 (0.45)**	**97.57 (0.45)**	**97.57 (0.45)**	99.46 (0.85)
14	**79.86 (2.10)**	79.41 (2.84)	78.80 (2.84)	77.71 (3.37)	77.71 (3.37)	78.01 (3.09)	78.73 (7.61)
15	97.43 (1.89)	95.30 (2.19)	**97.74 (1.69)**	95.66 (1.95)	74.28 (8.66)	97.10 (1.25)	98.44 (1.60)
16	**87.09 (4.35)**	86.57 (4.31)	86.87 (4.11)	86.31 (4.09)	84.30 (3.02)	85.67 (4.41)	89.89 (3.61)
17	**89.07 (3.66)**	88.58 (3.66)	88.79 (3.46)	87.90 (3.31)	85.74 (2.57)	87.37 (3.56)	91.13 (2.96)
18	**99.82 (0.80)**	**99.82 (0.80)**	**99.82 (0.80)**	**99.82 (0.80)**	**99.82 (0.80)**	**99.82 (0.80)**	99.99 (0.34)
19	72.87 (0.62)	**73.12 (0.62)**	72.89 (0.65)	72.50 (0.70)	72.84 (0.63)	72.64 (0.68)	73.73 (2.03)
20	51.06 (0.36)	51.06 (0.36)	**51.07 (0.35)**	51.06 (0.36)	51.03 (0.36)	51.04 (0.36)	51.12 (2.33)
21	**87.73 (0.81)**	87.46 (0.77)	86.48 (0.71)	86.90 (0.80)	87.21 (0.80)	86.77 (0.71)	89.16 (1.69)
22	41.98 (3.98)	43.18 (6.12)	**50.55 (2.04)**	45.85 (7.32)	33.67 (6.12)	42.78 (6.87)	74.09 (11.62)
23	**84.83 (1.26)**	84.82 (1.26)	84.82 (1.36)	84.64 (1.50)	83.20 (1.69)	84.61 (1.51)	84.72 (2.65)
24	**61.31 (2.48)**	60.72 (2.05)	60.88 (1.62)	57.48 (1.46)	57.68 (1.97)	60.59 (2.39)	66.15 (11.81)
25	62.98 (3.56)	63.26 (3.52)	62.33 (3.54)	63.22 (2.65)	**63.35 (4.39)**	62.07 (3.01)	67.08 (8.89)
26	64.92 (2.58)	**65.86 (2.45)**	64.87 (2.70)	62.81 (3.31)	65.23 (1.90)	64.74 (3.74)	57.67 (12.18)
27	62.83 (2.71)	61.80 (2.21)	**63.98 (2.48)**	60.19 (2.57)	58.82 (3.89)	61.62 (2.54)	62.96 (10.28)
28	60.14 (1.29)	**62.34 (2.12)**	61.43 (1.92)	58.90 (1.87)	58.05 (2.55)	60.19 (2.96)	64.59 (11.42)
29	57.45 (3.12)	57.95 (2.62)	57.81 (2.93)	57.53 (2.81)	57.67 (2.81)	**58.05 (2.89)**	53.78 (15.68)
30	48.39 (2.86)	**49.30 (3.41)**	49.21 (3.16)	49.04 (3.02)	48.19 (2.76)	49.04 (2.89)	52.22 (6.59)
31	69.62 (3.54)	69.52 (3.80)	71.45 (4.16)	69.63 (4.49)	72.06 (5.29)	**75.13 (5.51)**	60.89 (15.46)
32	55.01 (1.80)	53.93 (1.79)	55.02 (1.98)	53.38 (2.46)	53.15 (1.80)	**56.26 (2.45)**	53.31 (20.07)
33	57.85 (2.45)	58.75 (1.64)	57.29 (2.41)	58.07 (3.28)	50.58 (5.47)	**59.61 (2.13)**	57.66 (15.86)
34	**50.44 (2.75)**	**50.44 (2.45)**	49.68 (2.95)	49.39 (2.05)	48.99 (2.32)	49.46 (2.98)	56.29 (4.76)
Total Avg.	89.65 (0.17)	89.67 (0.17)	89.88 (0.17)	89.52 (0.17)	88.84 (0.19)	89.51 (0.17)	90.12 (0.16)

Table 12 – Relative query number (in percentage) of all Z values across all 34 datasets. The most reduced number of queries was marked in bold.

Datasets	Fixed Z					Z_MtL
	0.05	0.1	0.2	0.5	0.7	
1	**91.17**	**91.17**	**91.17**	**91.17**	**91.17**	**91.17**
2	**90.48**	**90.48**	**90.48**	**90.48**	**90.48**	**90.48**
3	**80.30**	**80.30**	**80.30**	**80.30**	**80.30**	**80.30**
4	**69.99**	**69.99**	**69.99**	**69.99**	**69.99**	**69.99**
5	**91.32**	**91.32**	**91.32**	**91.32**	**91.32**	**91.32**
6	92.62	90.40	**89.86**	**89.86**	**89.86**	**89.86**
7	**90.97**	**90.97**	**90.97**	**90.97**	**90.97**	**90.97**
8	72.54	**72.51**	**72.51**	**72.51**	**72.51**	**72.51**
9	**83.63**	**83.63**	**83.63**	**83.63**	**83.63**	**83.63**
10	**79.96**	**79.96**	**79.96**	**79.96**	**79.96**	**79.96**
11	**84.62**	**84.62**	**84.62**	**84.62**	**84.62**	**84.62**
12	70.04	**69.74**	**69.74**	**69.74**	**69.74**	**69.74**
13	**81.53**	**81.53**	**81.53**	**81.53**	**81.53**	**81.53**
14	75.79	49.04	34.60	1.69	**0.45**	21.32
15	7.02	4.92	5.79	3.87	**3.33**	4.48
16	56.99	52.19	41.18	24.45	**16.83**	20.44
17	50.88	47.50	35.11	18.16	**12.92**	15.27
18	**99.95**	**99.95**	**99.95**	**99.95**	**99.95**	**99.95**
19	91.53	94.38	89.84	60.95	59.14	**58.17**
20	99.94	99.93	99.91	98.70	**98.36**	99.37
21	91.57	88.49	77.75	56.48	**46.08**	54.00
22	6.52	5.81	14.04	3.74	5.54	**2.21**
23	95.41	93.24	84.79	38.38	**28.16**	36.01
24	70.88	66.05	67.84	41.36	**28.76**	38.44
25	80.14	78.81	79.29	65.38	53.82	**45.91**
26	56.09	52.11	47.69	30.34	31.16	**19.94**
27	81.99	70.51	76.14	65.38	**47.68**	56.56
28	80.29	77.46	79.44	51.16	49.85	**48.87**
29	44.22	42.51	35.41	27.36	**20.77**	25.29
30	56.42	51.43	49.08	38.06	**30.87**	36.58
31	29.88	25.04	21.70	15.52	**11.30**	11.88
32	48.70	44.12	34.12	24.06	**17.66**	20.32
33	51.67	45.22	36.52	27.02	16.99	**16.56**
34	85.64	77.91	79.04	70.34	**49.10**	67.57
Total Avg.	78.68	77.80	77.22	73.06	**71.24**	72.08

WORKS PUBLISHED BY THE AUTHOR

Published Works

Conferences

1. Vinicius Eiji Martins, Victor G Turrisi da Costa, Sylvio Barbon Junior. Active Learning Embedded in Incremental Decision Trees. In *Brazilian Conference on Intelligent Systems*, pages 367-381. IEEE, 2020.