Machine Learning (2023) 112:4351–4378
https://doi.org/10.1007/s10994-022-06188-7

STUDD: a student–teacher method for unsupervised concept drift detection

Vitor Cerqueira[1] · Heitor Murilo Gomes[2] · Albert Bifet[2,3] · Luis Torgo[1]

Received: 27 February 2021 / Revised: 28 January 2022 / Accepted: 9 May 2022 /
Published online: 21 June 2022

Abstract
Concept drift detection is a crucial task in data stream evolving environments. Most of state of the art approaches designed to tackle this problem monitor the loss of predictive models. However, this approach falls short in many real-world scenarios, where the true labels are not readily available to compute the loss. In this context, there is increasing attention to approaches that perform concept drift detection in an unsupervised manner, i.e., without access to the true labels after the model is deployed. We propose a novel approach to unsupervised concept drift detection based on a student-teacher learning paradigm. Essentially, we create an auxiliary model (student) to mimic the primary model's behaviour (teacher). At run-time, our approach is to use the teacher for predicting new instances and monitoring the *mimicking* loss of the student for concept drift detection. In a set of experiments using 19 data streams, we show that the proposed approach can detect concept drift and present a competitive behaviour relative to the state of the art approaches.

Keywords Concept drift detection · Data streams · Model compression

1 Introduction

Learning from time-dependent data is a challenging task due to the uncertainty about the dynamics of real-world environments. When predictive models are deployed in environments susceptible to changes, they must detect these changes and adapt themselves accordingly. The phenomenon in which the data distribution evolves is referred to as *concept drift*, and a sizeable amount of literature has been devoted to it Gama et al. (2014).

An archetype of concept drift is the interest of users in a service, which typically changes over time (Kim & Park, 2017). Changes in the environment have a potentially

Editors: Annalisa Appice, Grigorios Tsoumakas.

✉ Vitor Cerqueira
vitor.cerqueira@dal.ca

[1] Dalhousie University, Halifax, Canada

[2] University of Waikato, Hamilton, New Zealand

[3] Télécom ParisTech, Paris, France

strong negative impact on the performance of models (Gama et al., 2014). Therefore, it is fundamental that these models can cope with concept drift. That is, to detect changes and adapt to them accordingly.

Following Gama et al. (2014), concept drift can be split into two types: real concept drift and virtual concept drift. The former denotes changes in the conditional distribution of the target variable given the input explanatory variables—in this scenario, the marginal distribution of the explanatory variable does not have to change. The latter occurs if the marginal distribution of the input explanatory variables changes but not the conditional distribution.

Concept drift detection and adaptation are typically achieved by coupling predictive models with a change detection mechanism (Gomes et al., 2019). The detection algorithm launches an alarm when it identifies a change in the data. Typical concept drift strategies are based on sequential analysis (Page, 1954), statistical process control (Gama et al., 2004), or monitoring of distributions (Bifet & Gavalda, 2007). When change is detected, the predictive model adapts by updating its knowledge with recent information. A simple example of an adaptation mechanism is to discard the current model and train a new one from scratch. Incremental approaches are also widely used (Gomes et al., 2017).

The input data for the majority of the existing drift detection algorithms is the performance of the predictive model over time, such as the error rate. In many of these detection methods, alarms are signalled if the performance decreases significantly. However, in several real-world scenarios, labels are not readily available to estimate the performance of models. Some labels might arrive with a delay or not arrive at all due to labeling costs. This is a major challenge for learning algorithms that rely on concept drift detection as the unavailability of the labels precludes their application (Gomes et al., 2019).

In this context, there is increasing attention toward unsupervised approaches to concept drift detection. These assume that, after an initial fit of the model, no further labels are available during the deployment of this model in a test set. Most works in the literature handle this problem using statistical hypothesis tests, such as the Kolmogorov–Smirnov test. These tests are applied to the output of the models (Žliobaite, 2010), either the final decision or the predicted probability, or the input attributes (dos Reis et al., 2016).

Our goal in this paper is to address concept drift detection in an unsupervised manner. To accomplish this, we propose a novel approach to tackle this problem using a student–teacher learning paradigm called `STUDD` (**S**tudent–**T**eacher approach for **U**nsupervised **D**rift **D**etection). The gist of the idea is as follows. On top of the main predictive model, which we designate as the teacher, we also build a second predictive model, the student. Following the literature on model compression (Buciluă et al., 2006) and knowledge distillation (Hinton et al., 2015), the student model is designed to mimic the behaviour of the teacher.

Using the student–teacher framework, our approach to unsupervised concept drift detection is carried out by monitoring the student's mimicking loss. The mimicking loss is a function of the discrepancy between the teacher's prediction and student's prediction in the same instance. In summary, we use the student model's loss as a surrogate for the behaviour of the main model. Accordingly, we can apply any state of the art approach in the literature, which considers the loss of a model as the main input, for example, the Page-Hinkley test (Page, 1954).

When concept drift occurs, it causes changes in the classes' prior probabilities or changes in the class conditional probabilities of the explanatory variables. In effect, we hypothesise that, if these changes affect the distribution of the input explanatory variables (virtual concept drift), they will disrupt the collective behaviour between the teacher and student models. In turn, this change of behaviour may be captured by monitoring the mimicking loss of the student model.

We compared `STUDD` to several state-of-the-art methods, both unsupervised and supervised ones using 19 benchmark data streams. The results indicate that the proposed method is useful for capturing concept drift. `STUDD` shows a more conservative behaviour relative to other approaches, which is beneficial in many domains of application.

To summarise, the contributions of this paper are the following:

- `STUDD`: a novel method for unsupervised concept drift detection based on a student–teacher learning approach;
- A set of experiments used to validate the proposed method. These include comparisons with state of the art approaches, and an analysis of the different scenarios regarding label availability.

The proposed method is publicly available online.[1] Our implementation is written in Python and is based on the scikit-multiflow framework (Montiel et al., 2018). We also remark that this article is an extension of a preliminary work published previously (Cerqueira et al., 2020). The experimental setting has been extended considerably. While in the previous work we validated `STUDD` using synthetic drifts based on two data sets, we now focus on a realistic evaluation scenario based on 19 benchmark data streams. We also include an increased number of state of the art approaches in the experiments.

The rest of this paper is organised as follows. In the next section (Sect. 2), we formally define the problem of concept drift detection in data streams, while in the following section (Sect. 3), we briefly review the literature on the topic of our work. We describe the methodology behind `STUDD` in Sect. 4. The experiments are reported in Sect. 5. The results of these are discussed in Sect. 6. Finally, Sect. 7 concludes the paper.

2 Background

2.1 Problem definition

Let $D(X, y) = \{(X_1, y_1), \ldots, (X_t, y_t)\}$ denote a possibly infinite data stream, where each X is a q-dimensional array representing the input explanatory variables. Each y represents the corresponding output label. We assume that the values of y are categorical. The goal is to use this data set $\{X_i, y_i\}_1^t$ to create a classification model to approximate the function which maps the input X to the output y. Let $\mathcal{T}$ denote this classifier. The classifier $\mathcal{T}$ can be used to predict the labels of new observations X. We denote the prediction made by the classifier as $\hat{y}_{\mathcal{T}}$.

Many real-world scenarios exhibit a non-stationary nature. Often, the underlying process causing the observations changes in an unpredictable way, which degrades the performance of the classifier $\mathcal{T}$. Let $p(X, y)$ denote the joint distribution of the explanatory variables X and the target variable y. According to Gama et al. (2014), concept drift occurs if $p(X, y)$ is different in two distinct points in time across the data stream. Changes in the joint probability can be caused by changes in $p(X)$, the distribution of the explanatory variables

[1] https://github.com/vcerqueira/studd.

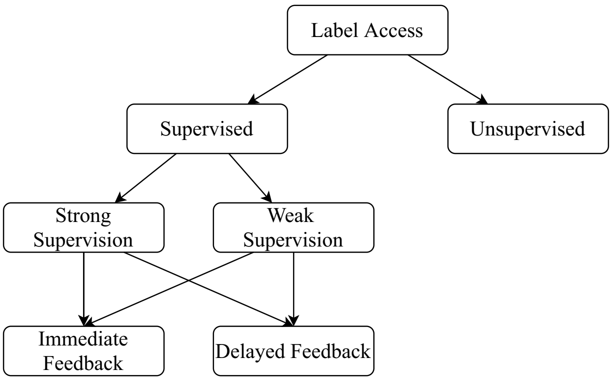

Fig. 1 The distinct potential scenarios regarding label access after the initial fit of the model (adapted from Gomes et al., 2017)

or changes in the class conditional probabilities $p(X|y)$ (Gao et al., 2007). These may eventually affect the posterior probabilities of classes $p(y|X)$. Depending on which distributions are affected, concept drift may be considered one of two types: real concept drift, or virtual concept drift. We described these in the previous section.

2.2 Label availability

When concept drift occurs, the changes need to be captured as soon as possible, so the decision rules of $\mathcal{T}$ can be updated. The vast majority of concept drift detection approaches in the literature focus on tracking the predictive performance of the model. If the performance degrades significantly, an alarm is launched and the learning system adapts to these changes.

The problem with these approaches is that they assume that the true labels are readily available after prediction. In reality, this is rarely the case. In many real-world scenarios, labels can take too long to be available, if ever. If labels do eventually become available, often we only have access to a part of them. This is due to, for example, labeling costs. The different potential scenarios when running a predictive model are depicted in Fig. 1.

Precisely, a predictive model is built using an initial batch of training data of which labels are available. When this model is deployed in a test set, concept drift detection is carried out in an unsupervised or supervised manner.

In unsupervised scenarios, no further labels are available to the predictive model. Concept drift detection must be carried out using a different strategy other than monitoring the loss. For example, one can track the output probability of the models (Žliobaite, 2010) or the unconditional probability distribution $p(X)$ (Kuncheva, 2004).

Concept drift detectors have access to labels when the scenario is supervised. On the one hand, the setting may be either strongly supervised or weakly supervised (Zhou, 2018). In the former, all labels become available. In the latter, the learning system only has access to a part of the labels. This is common in applications which data labeling is costly. On the other hand, labels can arrive immediately after prediction, or they can arrive with some delay. In some domains, this delay may be too large, and unsupervised approaches need to be adopted.

In this paper, we address concept drift detection from an unsupervised perspective. In this setting, we are restricted to use $p(X)$ to detect changes, as the probability of the explanatory variables is not conditioned on y.

3 Related research

In this section, we briefly review previous research related to our work. We split this review into two parts. In the first part, we overview approaches for concept drift detection, giving particular emphasis to unsupervised approaches. The second part addresses model compression and the related work on the student–teacher learning approach, which is the basis of the proposed method.

3.1 Concept drift detection

Concept drift can occur in mainly three different manners: suddenly, in which the current concept is abruptly replaced by a new one; gradually, when the current concept slowly fades; and reoccurring, in which different concepts are prevalent in distinct time intervals (for example, due to seasonality). A variation of gradual concept drifts are incremental drifts, which are extremely difficult to detect as they consist of many concepts that continually evolve.

We split concept drift detection into two dimensions: supervised and unsupervised. The supervised type of approaches assumes that the true labels of observations are available after prediction. Hence, they use the error of the model as the main input to their detection mechanism. On the other hand, unsupervised approaches preclude the use of the labels in their techniques.

3.1.1 Supervised approaches

Plenty of error-based approaches have been developed for concept drift detection. These usually follow one of three sort of strategies: sequential analysis, such as the Page-Hinkley test (PHT) (Page, 1954); statistical process control, for example the Drift Detection Method (DDM) (Gama et al., 2004) or the Early Drift Detection Method (EDDM) (Baena-Garcıa et al., 2006); and distribution monitoring, for example the Adaptive Windowing (ADWIN) approach (Bifet & Gavalda, 2007).

3.1.2 Unsupervised approaches

Although the literature is scarce, there is an increasing interest in approaches which try to detect drift without access to the true labels. Žliobaite (2010) presents a work of this type. She proposed the application of statistical hypothesis testing to the output of the classifier (either the probabilities or the final categorical decision). The idea is to monitor two samples of one of these signals. One sample serves as the reference window, while the other represents the detection window. When there is a statistical difference between these, an alarm is triggered. This process can be carried out using a sliding reference window (c.f. Fig. 2) or a fixed reference window (c.f. Fig. 3). The sliding reference window is more suitable for detecting drastic and abrupt data changes as it compares a window of instances adjacent to the detection window. Conversely, the fixed

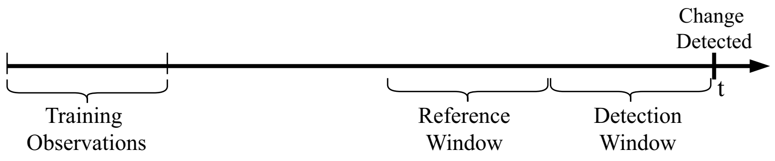

Fig. 2 Detecting changes using a sliding reference window. Change occurs at time t if the reference window is statistically different than the detection window

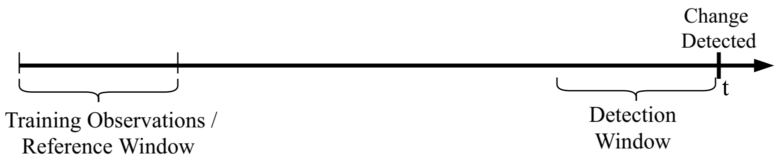

Fig. 3 Detecting changes using a fixed reference window

reference window may be fitting to detect gradual changes in the data, which are not apparent when the reference and detection window are adjacent.

In a set of experiments, Žliobaite shows that concept drift is detectable using this framework. The hypothesis tests used in the experiments are the two-sample Kolmogorov–Smirnov test, the Wilcoxon rank-sum test, and the two-sample t-test.

dos Reis et al. (2016) follow a strategy similar to Žliobaite (2010). They propose an incremental version of the Kolmogorov–Smirnov test and use this method to detect changes without using any true labels. However, they focus on tracking the attributes rather than the output of the predictive model. Specifically, they use a fixed window approach (c.f. Fig. 3) to monitor the distribution of each attribute. If a change is detected in any of these attributes, a signal for concept drift is issued.

In the same line of research, Yu et al. (2018) apply two layers of hypothesis testing hierarchically. Kim and Park (2017) also apply a windowing approach. Rather than monitoring the output probability of the classifier, they use a confidence measure as the input to drift detectors.

Pinto et al. (2019) present an automatic framework for monitoring the performance of predictive models. Similarly to the above-mentioned works, they perform concept drift detection based on a windowing approach. The signal used to detect drift is computed according to a mutual information metric, namely the Jensen–Shannon Divergence (Lin, 1991). The window sizes and threshold above which an alarm is launched is analysed, and the approach is validated in real-world data sets. The interesting part of the approach by Pinto et al. (2019) is that their method explains the alarms. This explanation is based on an auxiliary binary classification model. The goal of applying this model is to rank the events that occurred in the detection window according to how these relate to the alarm. These explanations may be crucial in sensitive applications which require transparent models.

Gözüaçık et al. (2019) also develop an auxiliary predictive model for unsupervised concept drift detection, which is called D3 (for Discriminative Drift Detector). The difference to the work by Pinto et al. (2019) is that they use this model for detecting concept drift rather than explaining the alarms.

3.2 Student–teacher learning approach

Model compression, also referred to as student-teacher learning, is a technique proposed by Buciluǎ et al. (2006). The goal is to train a model, designated as a student, to mimic the behaviour of the first model (the teacher). To perform model compression, the idea is to first retrieve the predictions of the teacher in observations not used for training (e.g. a validation data set). Then, the student model is trained using this set of observations, where the explanatory variables are the original ones, but the original target variable is replaced with the predictions of the teacher. The authors use this approach to compress a large ensemble (the teacher) into a compact predictive model (the student).

Buciluǎ et al. (2006) use the ensemble selection algorithm (Caruana et al., 2004) as the teacher and a neural network as the student model and address eight binary classification problems. Their results show that the compressed neural network performs comparably with the teacher while being “1000 times smaller and 1000 times faster”. Moreover, the compressed neural network considerably outperforms the best individual model in the ensemble used as the teacher.

Hinton et al. (2015) developed the idea of model compression further, denoting their compression technique as knowledge distillation. Distillation works by softening the probability distribution over classes in the softmax output layer of a neural network. The authors address an automatic speech recognition problem by distilling an ensemble of deep neural networks into a single and smaller deep neural network.

Both Buciluǎ et al. (2006) and Hinton et al. (2015), show that combining the predictions of the ensemble leads to a comparable performance relative to a single compressed model.

While our concerns are not about decreasing the computational costs of a model, we can leverage model compression approaches to tackle the problem of concept drift detection. Particularly, by creating a student model which mimics the behaviour of a classifier, we can perform concept drift detection using the loss of the student model. Since this loss is not conditioned on the target variable y, concept drift detection is carried out in an unsupervised manner.

4 Methodology

In this section, we describe `STUDD`, the proposed approach to unsupervised concept drift detection. `STUDD` is split into two steps: an initial offline stage, which occurs during the training of the data stream classifier (Sect. 4.1); and an online stage, when the method is applied for change detection (Sect. 4.2). In Sect. 4.3, we overview our adaptation approach after concept drift is detected.

4.1 Stage 1: student–teacher training

The first stage of the proposed approach refers to the training of the predictive models. This process is illustrated in Fig. 4. A batch of training observations is retrieved from the source data stream $\mathcal{D}$. These observations ($\mathcal{D}(X_{tr}, y_{tr})$) are used to train the classifier $\mathcal{T}$. This is the predictive model to be deployed in the data stream.

After creating $\mathcal{T}$, we carry out a student–teacher approach in which $\mathcal{T}$ acts as the teacher. First, $\mathcal{T}$ is used to make predictions on the training set. This leads to a new training

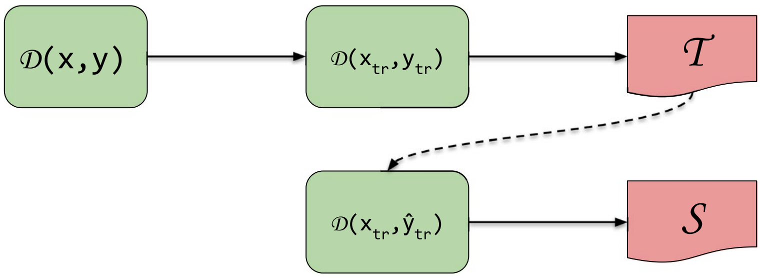
$\mathcal{D}(x,y)$
$\mathcal{D}(x_{tr},y_{tr})$
$\mathcal{T}$
$\mathcal{D}(x_{tr},\hat{y}_{tr})$
$\mathcal{S}$

Fig. 4 Fitting the teacher ($\mathcal{T}$) and student ($\mathcal{S}$) models using an initial batch of training observations

data set, in which the targets y_{tr} are replaced with the predictions of $\mathcal{T}$, $\hat{y}_{\{\mathcal{T},tr\}}$. Finally, the student model $\mathcal{S}$ is trained using the new data set. Essentially, the student model $\mathcal{S}$ is designed to mimic the behavior of the teacher $\mathcal{T}$.

It might be argued that using the same instances to train both the teacher and the student models leads to over-fitting. However, Hinton et al. (2015) show that this is not a concern.

The student-teacher learning paradigm is at the core of model compression (Buciluă et al., 2006) or knowledge distillation (Hinton et al., 2015) methods. These approaches aim at compressing a model with a large number of parameters (teacher), such as an ensemble or a deep neural network, into a more compact model (student) with a comparable predictive performance. Accordingly, the student model is deployed in the test set, while the teacher is not used in practice due to high computational costs.

Conversely, our objective for using a student-teacher strategy is different. We regard the student model $\mathcal{S}$ as a model which is able to predict the behavior of the teacher model $\mathcal{T}$, i.e., what the output of $\mathcal{T}$ will be for a given input observation. Moreover, it is important to remark that, in our methodology, both student and teacher models are applied in the test phase.

4.2 Stage 2: change detection

The second stage of the proposed method refers to the change detection process. As we have described before, the state-of-the-art concept drift detection methods take the loss of predictive models as their primary input. Since we assume that labels are unavailable, we cannot compute the model's loss $\mathcal{T}$. This precludes the typical application of state-of-the-art change detection approaches to unsupervised concept drift detection.

However, we can compute the student model's loss, which is independent of the true labels. The loss of the student is quantified according to the discrepancy between the prediction of $\mathcal{T}$ ($\hat{y}_{\mathcal{T}}$) and the prediction of $\mathcal{S}$ about $\hat{y}_{\mathcal{T}}$ ($\hat{y}_{\mathcal{S}}$). Accordingly, the loss of $\mathcal{S}$ is defined as $L(\hat{y}_{\mathcal{T}}, \hat{y}_{\mathcal{S}})$, where L is the loss function (e.g. the error rate).

Therefore, our approach to concept drift detection uses a state of the art detector, such as the Page-Hinkley test (Page, 1954). However, the main input to this detector is the student model's error, rather than the teacher model's error. This process is depicted in Fig. 5. For a given input observation x_i, we obtain the prediction from the models $\mathcal{T}$ and $\mathcal{S}$. Then, a function of the discrepancy between these predictions is given as input to the detection model.

When concept drift occurs, it potentially causes changes in the posterior probability of classes, $p(y|X)$. Thus, we hypothesise that such changes will also potentially affect the joint

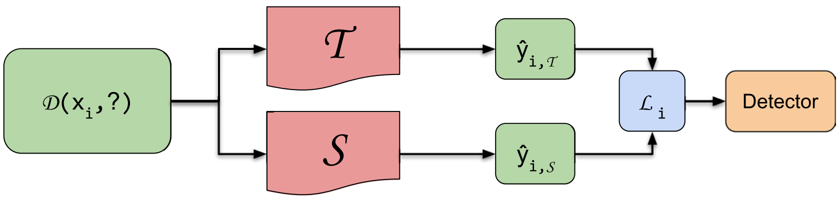

Fig. 5 The concept drift detection process of STUDD. We retrieve the predictions from both models for a new observation. A function of the discrepancy of these predictions, which is independent from the true labels, is given as input to a state of the art detection model

behaviour between student and teacher models. This effect will then be reflected on the student's imitation error, and the underlying change detection mechanism can capture it.

In effect, the teacher model is deployed in the data stream and used to make predictions on the upcoming observations. For concept drift detection, we track the error of the student model.

4.3 Concept drift adaptation

When a detection mechanism detects a change it triggers an adaptation process for the predictive model. In our work, we follow a retraining approach, in which the predictive model is retrained with a batch of recent observations. A common alternative to this is an incremental strategy in which predictive models are updated as labels become available.

5 Empirical experiments

This section details the experiments carried out to validate the proposed approach to unsupervised concept drift detection. We start by describing the research questions we aim at answering 5.1, followed by a brief description of the data streams used in the experiments 5.2. Afterwards, we explain the workflow used to analyse each approach under comparison 5.3, and in the respective evaluation scheme 5.4. We also describe the methods used to compare STUDD with 5.5, and detail the value of important parameters 5.7. Finally, the results are presented in Sect. 5.8.

5.1 Research questions

We designed a set of experiments to answer the following research questions:

- **RQ1**: Is STUDD able to detect concept drift?
- **RQ2**: What is the performance of STUDD for concept drift detection relative to state of the art approaches? These include both unsupervised and supervised ones;
- **RQ3**: When, in terms of label availability scenarios, is STUDD beneficial relative to a supervised approach?

5.2 Data sets

We used 19 benchmark data streams to answer the above research questions and validate the applicability of STUDD. These data sets include the following data streams: Electricity (Harries & Wales, 1999), forest cover type (Blackard & Dean, 1999), Poker (Cattral et al., 2002), Gas (Vergara et al., 2012), Luxembourg (Žliobaitė, 2011), Ozone (Dua & Graff, 2017), Power supply (Zhu, 2010), Rialto (Losing et al., 2016), Outdoor (Losing et al., 2015), Keystroke (Souza et al., 2015), NOAA (Ditzler and Polikar 2012), Bike (Fanaee-T & Gama, 2014), Arabic (Hammami & Bedda, 2010), Arabic with shuffled observations as per dos Reis et al. (2016), Insects (de Souza et al., 2013), Insects with artificial abrupt concept drift (dos Reis et al., 2016), Posture (Kaluža et al., 2010), and GMSC (Gomes et al., 2017). These are briefly described in Table 1. In order to speed up computations, we truncated the sample size of all data streams to 150.000 observations. These data sets are commonly used as benchmarks for mining data streams. We retrieved them from an online repository for data streams (Souza et al., 2020), or the repository associated with two previous works related to data streams.[2,3]

5.3 Workflow of experiments

We designed the experiments according to a batch setup, split into an offline stage and an online stage.

In the offline stage, we train the main classifier $\mathcal{T}$ to be deployed in the data stream using an initial batch of W observations. We also carry out any task specific to the underlying drift detection approach. For example, in the case of the proposed approach, we also train the student model $\mathcal{S}$.

The online stage starts when the classifier $\mathcal{T}$ is deployed in the data stream. For each new observation x_i, the classifier $\mathcal{T}$ makes a prediction $\hat{y}_i$. Meanwhile, the underlying detection mechanism uses the available data (e.g. x_i, $\hat{y}_i$) to monitor the classifier's behaviour. If the detection mechanism detects a change, it launches an alarm and the classifier $\mathcal{T}$ is adapted with recent information.

The adaptation mechanism adopted in this work is based on a re-training procedure. The current model is discarded, and a new model is re-trained using the latest W observations. This workflow is depicted in Fig. 6. We remark that, in the case of STUDD, the student model is also updated as described.

5.4 Evaluation

Our goal is to evaluate the performance of the concept drift detection mechanisms. We focus on unsupervised scenarios, in which the true labels are not readily available. However, for evaluation purposes, we use the labels to assess the quality of change detectors. We aim at measuring a trade-off between predictive performance and the number of alarms issued by the detector. The alarms represent a cost as they trigger the retrieval (and annotation) of a batch of observations.

We evaluate each approach from two dimensions:

[2] https://github.com/denismr/incremental-ks.

[3] https://github.com/hmgomes/AdaptiveRandomForest.

Table 1 Data streams used in the experiments

Data stream	Description	Shape	# Classes
Electricity	Price direction of electricity market in Australia	45.312×9	2
CoverType	Forest cover type from the US Forest Service	581.012×55	5
Poker	Poker hands drawn from a deck of 52 cards	$1.025.010 \times 12$	10
Gas	Gas measurements from chemical sensors	13.910×17	6
Luxembourg	Survey concerning the internet usage (high or low) from 2002 to 2007	1.901×31	2
Ozone	Air measurements concerning ozone levels	2.574×32	2
Sensors	Sensor identification from environmental data	$2.219.803 \times 5$	54
Powersupply	Hour identification from power supply data from an Italian electricity company	29.928×2	24
Rialto	Building identification near from processed images taken in the Rialto bridge in Venice	82.250×27	10
Outdoor	Object identification from images taken outdoor under varying lighting conditions (sunny and cloudy)	4.000×21	40
Keystroke	User identification from typing rhythm of an expression	1.600×10	4
NOAA	Rain detection from weather measurements collected over 50 years	18.159×8	2
Bike	Count of rental bikes (high or low) from a bike-sharing system	17.378×5	2
Arabic	Digit identification from audio (in arabic) features	$8.800 \times\times 28$	10
ArabicShuffled	Similar to *Arabic* but observations are shuffled by gender to enhance concept drift (c.f. dos Reis et al., 2016)	8.800×28	10
Insects	Identification of the specimen of a flying insect that is passing through a laser	5.325×50	5
InsectsAbrupt	Similar to *Insects*, but abrupt drift is introduced in the feature space	5.325×50	5
Posture	Movement identification from sensors carried by different people	164.859×4	11
GMSC	Credit scoring data set (*Give me some credit*)	150.000×11	2

The shape column describes the dimensionality of the data in the form (number of rows × number of columns)

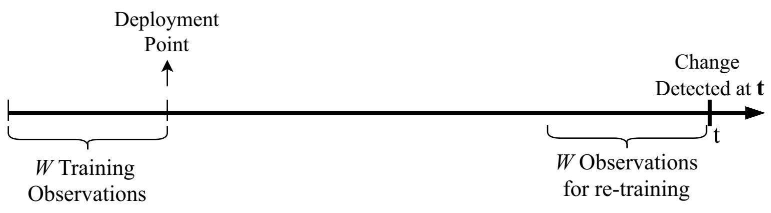

Fig. 6 The workflow for applying and evaluating each method under comparison

- Predictive performance: We measure the quality of the classifier according to the Cohen's Kappa statistic (Cohen, 1960), which is a common metric used to evaluate classification models;
- Annotation costs: We assume that we are working in environments where labels are scarce and costly to obtain. Therefore, as explained before, each concept drift signal triggers a request for an additional batch of labels. This request is expensive to the user, and an important metric to minimize. We account for this problem by measuring the ratio of labels with respect to the complete length of the data stream used by the respective approach.

Ideally, the optimal approach maximizes predictive performance and minimizes the amount of labels requested.

5.5 Methods

Besides the proposed method we include nine other approaches in our experimental setup. These can be described as follows.

We include the following two baselines:

- `BL-st`: A static baseline, which never adapts to concept drift. In practice, a model (Random Forest) is fit using the training observations and used to predict the subsequent ones remaining in the data stream. Accordingly, the model may become outdated due to concept drift, but incurs minimal labeling costs;
- `BL-ret`: A baseline which follows the opposite strategy to `BL-st`; it retrains the predictive model after every W observations. The model is always up to date, but at the price of high labeling costs.

In terms of unsupervised methods, which do not use any true labels, we apply the following state of the art approaches:

- `Output Sliding` (`OS`): A method that tracks the output of the predictive model using a **sliding** reference window as described by Žliobaite (2010). Figure 2 shows the workflow of this approach. According to previous studies (Žliobaite, 2010; dos Reis et al., 2016), we apply the Kolmogorov–Smirnov test to assess whether or not change occurs (c.f. Sect. 3.1.2 for more details);
- `Output Fixed` (`OF`): A similar approach to `OS`, but which tracks the output of the predictive model using a **fixed** reference window. This approach is depicted in Fig. 3. We also apply the Kolmogorov–Smirnov test in this case.

- `Feature Fixed` (`FF`): The method described by dos Reis et al. (2016), which instead of tracking the output of predictive models (such as `OS` or `OF`), it tracks the values of features. Following dos Reis et al. (2016), if a change is detected in any of the features using the Kolmogorov–Smirnov test, the predictive model is adapted.

We also include the following supervised approaches in our experiments. These assume some level of access to the true labels. While they may not be applicable in some scenarios where labels are difficult to acquire, they are important benchmarks for comparisons.

- `Strongly Supervised` (`SS`): We apply the standard concept drift detection procedure which assumes that all the true labels are immediately available after making a prediction. This can be regarded as the gold standard. The term *strong* refers to the fact that all labels are available during testing (Zhou, 2018);
- `Weakly Supervised` (`WS`): In many real-world scenarios, particularly in high-frequency data streams, data labeling is costly. Hence, predictive models can only be updated using a part of the entire data set. This process is commonly referred to as weakly supervised learning (Zhou, 2018). We simulate a weakly supervised scenario in our experiments. Accordingly, predictive models only have access to *l_access%* of the labels. In other words, after a model predicts the label of a given instance, the respective label is immediately available with a *l_access%* probability;
- `Delayed Strongly Supervised` (`DSS`): Labels can take some time to be available. We study this aspect by artificially delaying the arrival of the labels by *l_delay* instances. After a label becomes available, the respective observation is used to update the change detection model;
- `Delayed Weakly Supervised` (`DWS`): We combine the two previous scenarios. In the `DWS` setup, only *l_access%* of the labels are available. Those which are available arrive with a delay of *l_delay* observations.

Note that all methods above follow the procedure outlined in Sect. 5.3. There are two differences between these approaches: (1) the degree of access to labels; and (2) how concept drift detection is carried out.

5.6 Learning algorithm

5.6.1 Assumptions

In terms of application, the proposed method is agnostic to the underlying predictive models. Any learning algorithm can be used to train the teacher and the student models, and these do not need to be the same. Notwithstanding, we assume that these are *strong* algorithms which model the input data with arbitrarily good predictive performance. Regarding the student, a strong model is important so it is able to model the input data along with the bias of the teacher model. A strong teacher model is also crucial for two main reasons. One is that in practice we want to maximize the predictive performance of the main model and solve the underlying task. The second reason, and more related to the proposed method, is that a weak model would be too stable to cause any significant effect on the error of the student. That is, the behavior of the teacher would be stable and predictable. Consider the extreme scenario in which the teacher model is the majority predictor, whose output is the

Table 2 Learning algorithm used to train the teacher and student models in each data set

Data	Teacher	Student
AbruptInsects	RF	RF
Insects	XGB	XGB
Posture	XGB	RF
Arabic	XGB	XGB
Bike	XGB	RF
NOAA	RF	RF
Sensor	DT	DT
Powersupply	SVC	RF
Poker	DT	DT
Rialto	XGB	XGB
Ozone	XGB	XGB
Outdoor	RF	RF
Luxembourg	RF	RF
Gas	RC	RC
Keystroke	XGB	RF
ArabicShuffled	RF	RF
Covtype	LR	LR
GMSC	XGB	XGB
Electricity	NB	XGB

majority class for all instances. In this case, the error of the student model would be constant zero, and the whole framework would fail.

5.6.2 Algorithm selection

In effect, we introduce an algorithm selection procedure in the experiments to pick the best possible models (teacher and student) for each input data set. We consider the following pool of learning algorithms:

RF	Random forest (Breiman, 2001)
SVM	Support vector machine (Chang & Lin, 2011)
DT	Decision tree (Breiman et al., 1984)
LR	Logistic regression (Friedman et al., 2001)
RC	Ridge classifier (Friedman et al., 2001)
NB	Naive Bayes (Friedman et al., 2001)
XGB	Extreme gradient boosting (Chen & Guestrin, 2016)

We resorted to the *scikit-learn* library (Pedregosa et al., 2011) for the implementation of these algorithms.

For each data set, we first select the best teacher algorithm, and then the corresponding student algorithm. We start by splitting an initial batch of W observations. These are split into a train and validation sets, in which the former contains the initial 70% instances. Then, each algorithm in the available pool is fit in the training data, and evaluated in the validation data. The selected algorithm is the one maximizing predictive performance

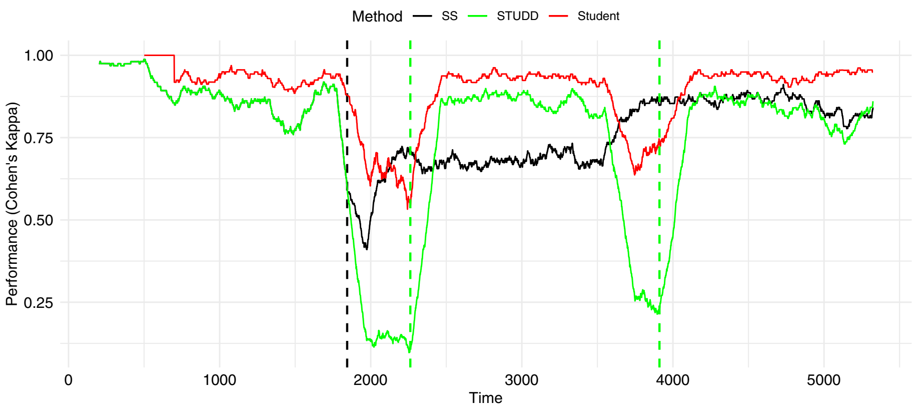

Fig. 7 An example using the *AbruptInsects* data stream where the proposed method is able to detect concept drift and adapt to the environment similarly to a supervised approach (Color figure online)

according to Cohen's Kappa statistic (Cohen, 1960). The selected algorithms for each data set are reported in Table 2. Note that all methods under comparison leverage this optimization process, though only `STUDD` applies the student model.

5.7 Parameter setup

In terms of parameters, we set the training window size W to 2000 observations for most data streams. Due to low sample size, for the data streams *Insects*, *AbruptInsects*, *Keystroke*, *Ozone*, *Outdoor*, and *Luxembourg*, we set this parameter to 500 observations. We follow the setup used by dos Reis et al. (2016) to set these values.

We apply the Page-Hinkley test (Page, 1954) for concept drift detection, specifically its implementation from the *scikit-multiflow* library (Montiel et al., 2018). This approach is a state of the art method for concept drift detection. We set the value of the δ parameter, which concerns the magnitude of changes, to 0.001, while the remaining parameters are left as default.

For the state of the art unsupervised concept drift detection approaches, the significance level parameter for rejecting the null hypothesis in the Kolmogorov–Smirnov test is set to 0.001, similarly to dos Reis et al. (2016). Regarding the delayed supervised methods (`DSS` and `DWS`), we set the delay parameter (*l_delay*) to $W/2$, which is half of the training window size. For the weakly supervised variants (`WS` and `DWS`), the access to labels (*l_access*) is set to 50. Finally, the loss function used as input to the Page-Hinkley test is the error rate.

5.8 Results

In this section, we present the results obtained from the experiments. First, we start by visualizing the alarms triggered by `STUDD` for concept drift and comparing to those of a supervised benchmark method (Sect. 5.8.1). Then, we present the main results which shows the performance of each approach and the respective costs (Sect. 5.8.2). Finally, we carry out a sensitivity analysis which compares the performance of `STUDD` with a supervised approach with varying degrees of access to labels (Sect. 5.8.5).

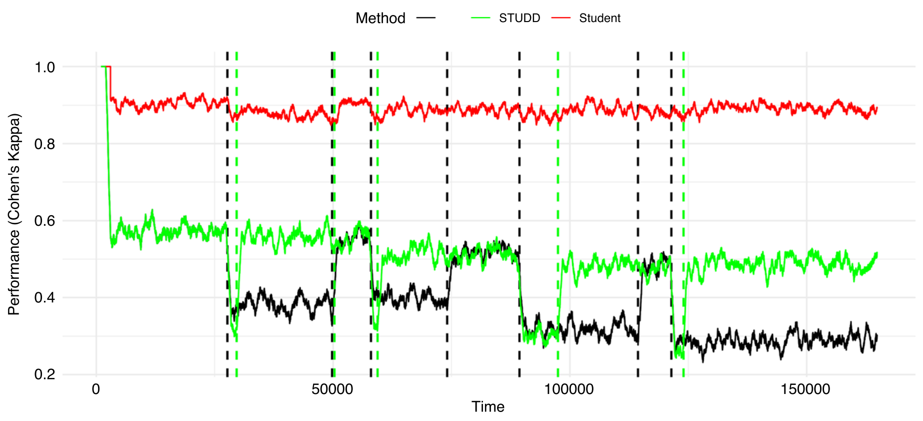

Fig. 8 An example using the *Posture* data stream where the proposed method is able to detect concept drift and adapt to the environment similarly to a supervised approach

5.8.1 Visualizing alarms

We start the analysis of the results by visualizing the alarms launched and the predictive performance by STUDD. We also include the behaviour of SS for a comparative analysis. In the interest of conciseness, we focus on three examples out of the 19 problems: two successful examples, in which STUDD is able to detect concept drift and obtain a competitive performance with a supervised approach with complete access to the true labels; and a negative example, which shows a problem in which STUDD performs poorly.

The first example is shown in Fig. 7. The figure shows the performance of SS, STUDD, and the student model of STUDD across the data stream InsectsAbrupt. The performance is computed in a sliding window of 200 observations. The vertical dashed lines represent the time points in which the respective approach (SS or STUDD) triggers an alarm for concept drift.

In the initial part of the data stream the performance of both approaches is identical (black and green lines are superimposed). Their behaviour are different from the point SS triggers the first alarm. This alarm has a visible impact on predictive performance because the score of STUDD continues to decreases considerably. Notwithstanding, STUDD is able to detect the change soon after and regain the previous level of predictive performance. The STUDD method behaves as expected. Initially, the student model is able to predict the predictions of the teacher with an arbitrarily good performance. However, when concept drift occurs, this performance decreases which leads to concept drift detection.

Figure 8 shows another example for the data stream *Posture* which follows the same structure as the previous one. In this case, STUDD is also able to launch an alarm soon after SS, and provide a competitive adaptation (despite launching fewer alarms) relative to the supervised approach. These examples show that the proposed approach is able to detect changes in the environment.

We show a final example in Fig. 9 for data stream *Bike*. In this scenario, STUDD struggles to adapt to the environment, though an early alarm is triggered. In the figure, it is visible that the performance of the teacher and student models are contrasting in someintervals of the data, for instance around data point 5000. While the performance score of the teacher is 0, which denotes a poor performance, the performance of the student is perfect

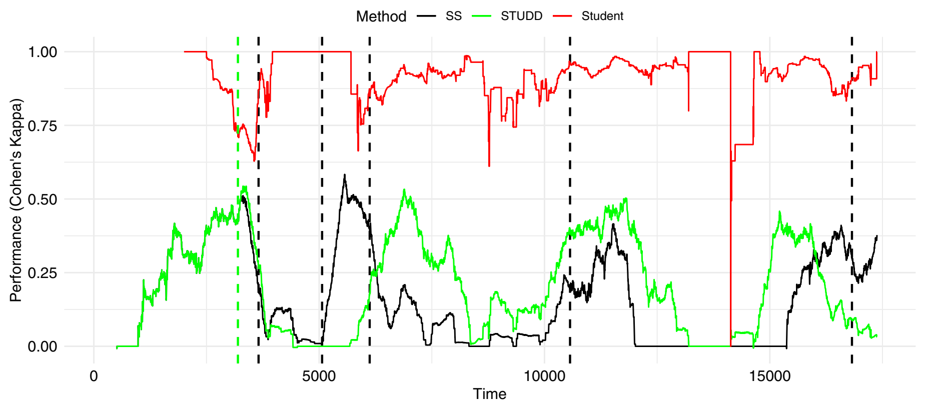

Fig. 9 An example using the *Bike* data stream where the proposed method struggles to detect changes in the environment (Color figure online)

Table 3 Performance of each method in each data set according to to Cohen's kappa score

	STUDD	BL-st	BL-ret	SS	DSS	WS	DWS	OS	OF	FF
AbruptInsects	0.73	0.59	0.74	0.78	0.78	0.79	0.73	0.79	0.80	0.77
Insects	0.81	0.81	0.77	0.74	0.74	0.79	0.81	0.71	0.70	0.77
Posture	0.50	0.34	0.51	0.40	0.48	0.46	0.50	0.49	0.47	0.49
Arabic	0.71	0.63	0.72	0.78	0.76	0.68	0.75	0.73	0.75	0.72
Bike	0.32	0.31	0.31	0.30	0.35	0.32	0.32	0.23	0.32	0.31
NOAA	0.39	0.39	0.45	0.44	0.43	0.46	0.41	0.46	0.40	0.46
Sensor	0.59	0.11	0.67	0.83	0.63	0.77	0.56	0.53	0.60	0.67
Powersupply	0.10	0.09	0.11	0.11	0.09	0.11	0.11	0.11	0.10	0.11
Poker	0.48	0.28	0.46	0.65	0.61	0.68	0.58	0.20	0.18	0.45
Rialto	0.38	0.30	0.34	0.47	0.37	0.40	0.36	0.40	0.36	0.34
Ozone	0.13	0.13	0.17	0.14	0.13	0.13	0.13	0.14	0.17	0.17
Outdoor	0.36	0.37	0.38	0.38	0.34	0.32	0.36	0.38	0.37	0.38
Luxembourg	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Gas	0.48	0.32	0.51	0.62	0.53	0.58	0.51	0.44	0.57	0.51
Keystroke	0.84	0.84	0.93	0.88	0.84	0.84	0.84	0.84	0.84	0.93
ArabicShuffled	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88
Covertype	0.52	0.50	0.67	0.63	0.58	0.53	0.53	0.33	0.54	0.66
GMSC	0.32	0.32	0.06	0.32	0.32	0.32	0.32	0.32	0.32	0.11
Electricity	0.29	0.13	0.39	0.39	0.39	0.39	0.34	0.33	0.38	0.39

(score of 1). Effectively, the student is able to anticipate the behaviour of the teacher model, though this behaviour is quite poor in terms of predictive ability. Consequently, no alarm is triggered for concept drift as the performance of the student model is good and stable. The main reason for this outcome is closely tied with the assumptions we put forth in Sect. 5.6.1. We hypothesized that STUDD is able to detect changes provided both teacher and student models have an arbitrarily good predictive performance.

Table 4 Ratio of additional labels (with respected to the full length of the data stream) required by each method in each data set

	STUDD	BL-st	BL-ret	SS	DSS	WS	DWS	OS	OF	FF
AbruptInsects	0.28	0.09	0.94	0.19	0.19	0.19	0.28	0.19	0.19	0.66
Insects	0.09	0.09	0.94	0.28	0.28	0.09	0.09	0.47	0.66	0.94
Posture	0.02	0.01	0.99	0.08	0.05	0.11	0.11	0.02	0.02	0.74
Arabic	0.23	0.11	0.91	0.34	0.23	0.80	0.57	0.34	0.23	0.80
Bike	0.06	0.06	0.98	0.46	0.35	0.46	0.17	0.06	0.06	0.98
NOAA	0.06	0.06	0.99	0.22	0.22	0.28	0.17	0.28	0.22	0.99
Sensor	0.16	0.01	0.99	1.43	0.51	2.05	0.71	0.37	0.47	0.99
Powersupply	0.07	0.03	0.97	0.30	0.10	0.33	0.20	0.43	0.57	0.94
Poker	0.01	0.01	0.97	2.69	2.51	4.51	3.45	0.43	0.62	0.97
Rialto	0.21	0.01	1.00	1.23	0.50	1.16	0.79	0.50	0.75	1.00
Ozone	0.39	0.39	1.18	0.39	0.39	0.39	0.39	0.39	0.39	0.99
Outdoor	0.25	0.25	1.00	0.50	0.62	0.25	0.62	0.62	0.75	1.00
Luxembourg	0.53	0.53	1.05	0.53	0.53	0.53	0.53	0.53	0.53	1.05
Gas	0.86	0.07	0.93	0.89	0.78	2.00	1.58	0.50	0.86	0.93
Keystroke	0.31	0.31	0.94	0.31	0.31	0.31	0.31	0.31	0.31	0.63
ArabicShuffled	0.11	0.11	0.91	0.11	0.11	0.11	0.11	0.23	0.11	0.23
Covtype	0.09	0.01	0.99	0.49	0.42	0.51	0.16	0.09	0.29	0.99
GMSC	0.01	0.01	0.99	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Electricity	0.43	0.02	0.99	0.69	0.51	0.82	0.29	0.09	0.38	0.99

The above examples show the behaviour of STUDD in different scenarios. In the next section, we will analyse its performance in all data streams and compare it to state of the art approaches.

5.8.2 Performance by data stream

The main results are presented in Tables 3 and 4 . The first one reports the Kappa score of each approach across each data set. The second table has a similar structure as Table 3, but the values represent the ratio of labels (with respect to the full length of the data stream) used by the respective approach.

We start by studying the results according to the average rank, both in terms of predictive performance and labeling costs. The average rank describes the average relative position of each method (the lower the better) under comparison across the data sets.

The analysis of the average rank is presented in Fig. 10, which shows a scatter plot that compares the average rank of each method both in terms of predictive performance (*x*-axis) and in terms of labeling costs (*y*-axis). Overall, there is a strong trend which indicates that a better score in one component leads to a worse score in the other.

The baseline methods BL-st and BL-ret represent two extremes: The former shows the worsts average rank in predictive performance but the best one in terms of costs, while the latter presents one of the best scores for predictive performance but at high costs relative the the other approaches. Every other method shows a trade-off between predictive performance and labeling costs: a better score in one component

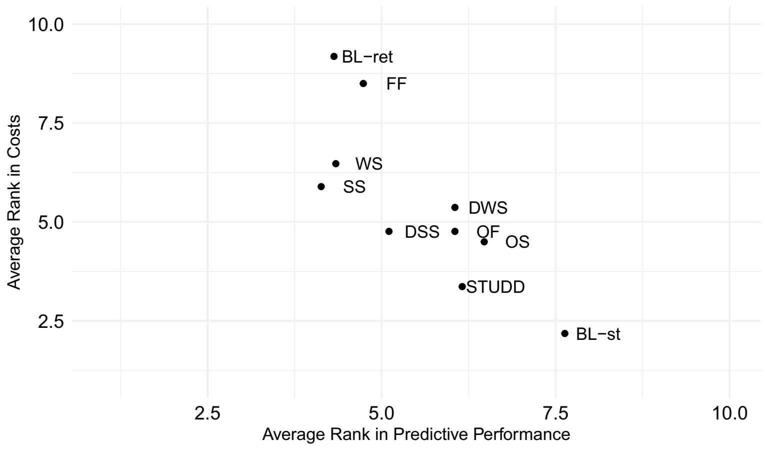

Fig. 10 Scatter-plot which compares the average rank of each model in terms of performance and the average rank of each model in terms of costs

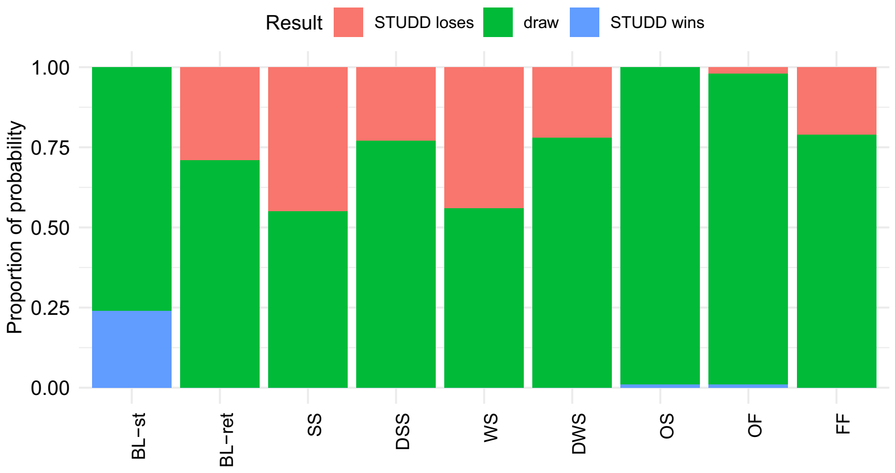

Fig. 11 Results of applying the Bayesian sign test to the percentage difference in predictive performance between STUDD and the respective method. The stacked bars represent the proportion of probability for each outcome

leads to a worse one in the other. Focusing on STUDD, it shows a comparable performance score relative to two other unsupervised approaches, namely OS and OF. The remaining method of this type (FF) shows the best score among unsupervised approaches, though it also shows considerable labeling costs relative to the others. In terms of costs, STUDD is the best non-baseline approach.

The average rank analysis only measures the relative position of each method (averaged across data sets), and it does not take into account the magnitude in the difference between scores. In many of the data sets, the difference of performance between STUDD and the other approaches is negligible while the difference in labeling costs (in favor of STUDD) are considerable. For example, in the *Rialto* dataset, STUDD shows a

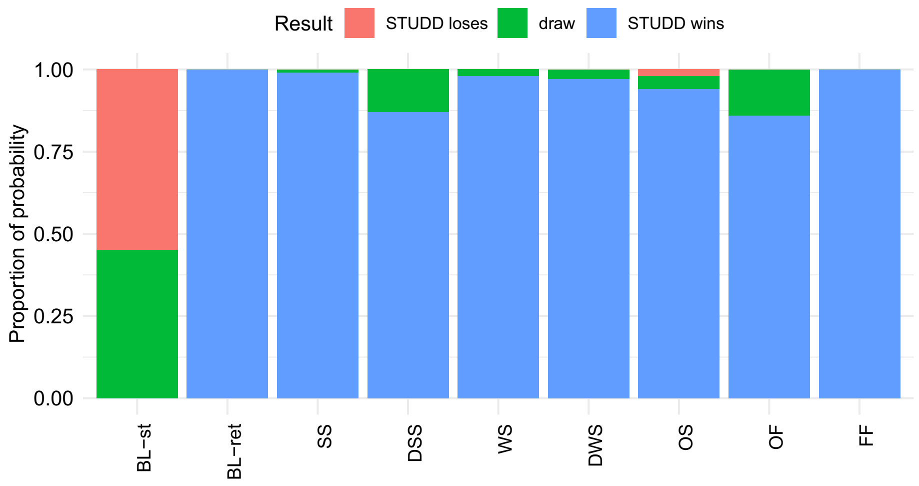

Fig. 12 Results of applying the Bayesian sign test to the percentage difference in labeling costs between `STUDD` and the respective method

competitive predictive performance (c.f. Table 3) with other approaches but with considerable less costs (c.f. Table 4). This outcome is also visible in other problems, such as *Powersupply*, *Outdoor*, among others.

We carried out a Bayesian analysis to assess the significance of the results in terms of magnitude of differences. We applied the Bayes sign test (Benavoli et al., 2014) to measure the significance of the percentage difference in scores between `STUDD` and each of the other approaches in the experiments. We applied this test with a ROPE (region of practical equivalence) value of 5%, which means that the respective pair of methods under comparison are practically equivalent if the absolute difference in their scores falls below this value. The results for predictive performance are shown in Fig. 11 while the results for labeling costs are presented in Fig. 12. We refer to Benavoli et al. (2017) for a comprehensive read on Bayesian analysis for comparing multiple approaches. As an example from Fig. 11, `STUDD` has around 25% probability of winning against the baseline `BL-st`, while with around 75% the outcome is a draw (i.e. percentage difference in predictive performance below 5%).

In terms of predictive performance, `STUDD` often leads to comparable scores relative to the remaining methods. When compared to supervised approaches (`SS`, `DSS`, `WS`, and `DWS`), `STUDD` either shows practically equivalent predictive performance or loses significantly. Relative to unsupervised approaches (`OS`, `OF`, and `FF`), `STUDD` shows an equivalent performance, though there is around 25% probability of losing to `FF`.

The main benefit of `STUDD` becomes apparent in Fig. 12, when the Bayesian sign test is applied to the labeling costs. Except for the baseline `BL-st` (which is never retrained), `STUDD` wins significantly (incurs significantly less costs) over all other methods with high probability. In summary, `STUDD` often shows a practically equivalent predictive performance but with significantly less labeling costs.

Table 5 Alarm analysis for each method across 16 datasets

Method	MDR	FAR	MTD	MTFA
`STUDD`	0.19	0	0.09	*NA*
`DSS`	0.06	0	0.14	*NA*
`WS`	0.12	0	0.14	*NA*
`DWS`	0.19	0	0.16	*NA*
`OS`	0.06	0.12	0.12	0.44
`OF`	0.06	0.12	0.11	0.30
`FF`	0.00	0.25	0.07	0.15

5.8.3 Analysing the alarms

As we described before, the results presented above evaluate each method by focusing on the trade-off between predictive performance and the costs of triggering an alarm (and acquiring labeled instances). In this section and for completeness, we also analyse the detection methods using typical drift detection metrics, namely:

- Missed detection ratio (MDR), which represents the ratio of datasets in which the respective approaches fail to detect concept drift;
- False alarm ratio (FAR), denoting the ratio of datasets in which the respective approaches launch at least one false alarm;
- Mean time to detection (MTD), which quantifies the average number of data points it takes for a method to detect a change—since we work with multiple data sets with distinct sample size we normalize MTD by the total number of observations of the data;
- Mean time between false alarms (MTFA), which denotes the mean time between false alarms triggered by the model—Similarly to MTD, this metric is also normalized by the sample size of the respective dataset.

These metrics are, to an extent, implicitly incorporated in the previous analysis. In principle, lower (better) scores in MDR, FAR, or MTD denote a more accurate detection and faster adaptation, which consequently leads to a better predictive performance. On the other hand, higher (better) MTFA scores represent less, in principle unnecessary, labeling costs.

To compute these metrics we have to know in advance at which time instance concept drift occurs. This point is difficult to find in practice, and the typical approach is to resort to synthetic experimental designs, e.g. (Žliobaite, 2010; Bifet, 2017). In our work, we keep the same experimental design presented in the previous sections, and use the behaviour of `SS` as benchmark. Essentially, we compute the metrics outlined above using the first alarm triggered by `SS` as the concept drift occurrence. Since the drift point is based on the alarms of a specific method, we apply the following approach for considering true detections and false alarms: if a given detector trigger an alarm up to W observations before the first alarm of `SS`, then the alarm is considered a true detection. Otherwise, if the alarm occurs more than W observations before the first alarm of `SS`, it is considered a false positive. We introduce this window of W observations because a detector may be faster to detect a change and should not be penalized for it.

The results for this analysis are presented in Table 5. We exclude the baselines from this study as they do not comprise a change detection mechanism. Moreover, the MTD and MTFA

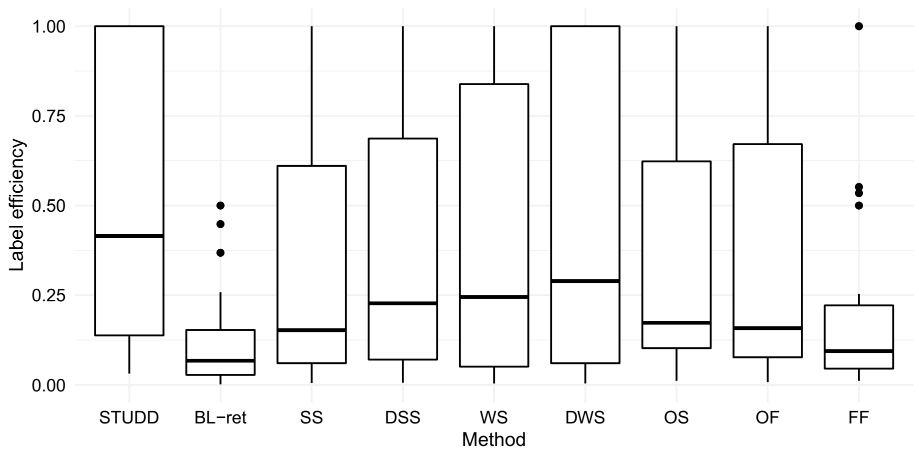

Fig. 13 Distribution of label efficiency of each method across the 19 datasets

scores are only computed for the 16 out of 19 datasets for which SS triggers at least one alarm. The three datasets excluded are the following: *Luxembourg*, *ArabicShuffled*, *GMSC*.

STUDD shows a MDR of around 19% (3 out of 16 problems), which is the highest value recorded (tied with DWS). We studied the outcome in these three particular data sets, which are the *Insects*, *NOAA*, and *Keystroke* problems. From the results in Table 3, STUDD leads to the best performance in one of the datasets (*Insects*), tied with DWS. In the *Keystroke* problem, only two approaches show a better predictive performance. In the remaining dataset, *NOAA*, STUDD shows the second to last performance. Overall, although STUDD does not trigger any alarm in these three problems, this results in a competitive performance in two cases.

5.8.4 Analysis of label efficiency

This section analyses the competing methods according to their labeling efficiency. Labeling efficiency combines the two metrics used before (predictive performance and annotation costs) into a single measure. It attempts to measure how much performance a given method gains by retrieving additional labels.

We define the labeling efficiency of a method m according to the ratio between its predictive performance and its annotation costs. These scores are normalized by those of the baseline BL-st in order to obtain a scale-independent metric. Label efficiency can be formalized as follows:

$$Normalized\_Performance(m) = \frac{Performance(m)}{Performance(\texttt{BL}-\texttt{st})} \tag{1}$$

$$Normalized\_Cost(m) = \frac{Cost(m)}{Cost(\texttt{BL}-\texttt{st})} \tag{2}$$

$$Labeling\_Efficiency(m) = \frac{Normalized\_Performance(m)}{Normalized\_Cost(\texttt{BL}-\texttt{st})} \tag{3}$$

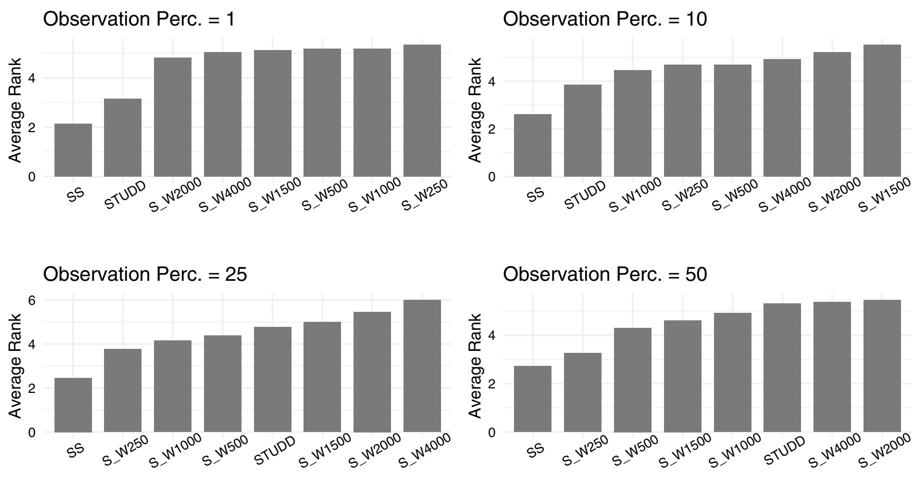

Fig. 14 Analysing the results for different values of *l_access* and *l_delay*. Each barplot represents the average rank of each approach across the 19 data streams

This metric assumes that the scores are non-negative.

Figure 13 shows the distribution of label efficiency of each method across the 19 datasets. The proposed approach `STUDD` shows the best overall label efficiency distribution. As previously mentioned, `STUDD` shows a comparable (and sometimes slightly worse) predictive performance relative to other approaches. However, the labeling efficiency is the best among the competing methods because of the significantly lower annotation costs. Excluding the baseline `BL-ret`, the unsupervised approach `FF` shows the worst labeling efficiency distribution. In effect, while `FF` shows one of the best predictive performance (the best among unsupervised approaches, including `STUDD`) its edge is accompanied by higher annotation costs.

5.8.5 Sensitivity analysis to label access and delay

In the previous sections we showed the applicability of `STUDD` for concept drift detection, and how it compares with other state of the art approaches. While we approach the concept drift task in a completely unsupervised manner, there may be scenarios in which labels are available, though in a limited manner. We described these scenarios in Sect. 2.2. We introduced observation delay and availability in some of the methods used in the experiments, namely `DSS`, `WS`, and `DWS`. In this section, we aim at making another comparison between `STUDD` and these methods. Specifically, our goal is to study the relative performance of these methods for different values of label delay (*l_delay*) and label access (*l_access*).

As described in Sect. 5.5, we define the label availability according to two parameters: *l_access%*, which denotes the probability of a label becoming available; and *l_delay*, which represents the number of observations it takes for a label to become available. For *l_access*, we test the following values: $\{1, 10, 25, 50\}$. In terms of *l_delay*, the set of possibilities is $\{250, 500, 1000, 1500, 2000, 4000\}$. For example, suppose that *l_access* is equal to 50 and *l_delay* is set to 250. This means that a label becomes available with 50% probability after 250 observations.

We show the results of this analysis in Fig. 14, which presents four barplots, one for each value of l_access. Each barplot represents the average rank of each method across the 19 data streams. A method has rank 1 in a data set if it presents the best performance score in that task. In each barplot, we include `SS`, `STUDD`, and six supervised variants (one for each delay value). Each one of these methods is identified by the delay value. For example, `S_W2000` represents a supervised variant with a delay of 2000 observations and the respective *l_access*. We note that in each barplot, the value of *l_access* is only valid for the six supervised variants and not for `SS` or `STUDD`.

The results show that `STUDD` performs relatively better as the probability of the label availability (*l_access*) decreases. Regarding the delay time (*l_delay*), lower values typically lead to better performance in terms of average rank. However, this parameter has a weaker impact relative to *l_access*. For example, for a *l_access* equal to 50, `STUDD` is the worst approach irrespective of the delay time. In summary, the results indicate that the proposed approach is beneficial if label acquisition is a problem.

6 Discussion

6.1 Main results

In the previous section we analysed the proposed approach for concept drift detection. `STUDD` is designed to detect changes in the environment without any access to true labels after the model is deployed. In this sense, we refer to this approach as unsupervised.

The results obtained provided enough empirical evidence verifying the ability of `STUDD` to detect concept drift (**RQ1**). While its predictive performance is comparable to other unsupervised approaches in most of the problems, it is often able to considerably reduce the label requirements (**RQ2**) as it triggers few alarms. This feature is important in domains of application in which the annotation process or false alarms are costly. We also compared `STUDD` with several variants of supervised approaches to concept drift detection. The results indicated that `STUDD` provides better performance only if the access to labels is low (**RQ3**).

6.2 Types of concept drift and label availability

In the introductory section of this paper we outlined the two main types of concept drift: real concept drift and virtual concept drift. Similarly to other unsupervised approaches to concept drift, the proposed methodology assumes that labels are unavailable. It is clear that these methods are only able to capture changes if they affect the input explanatory variables. Therefore, `STUDD` is appropriate for capturing virtual concept drift.

We believe that addressing this limitation represents an important direction for future research. `STUDD`, and other unsupervised approaches for concept drift, do not make use of labels even if they are available. Yet, while in many domains of applications labels are scarce they may be available in a small quantity (c.f. Fig. 1). Therefore, attempting to combine unsupervised approaches for concept drift with supervised approaches with potentially delayed or limited feedback may be worthwhile.

6.3 Choice of learning algorithm

We make an important assumption regarding the learning algorithms when applying STUDD. This method relies on predictive models (both teacher and student) which provide an arbitrary good fit to the data. Otherwise, the behaviour of the teacher may become too predictable, which leads to a stable performance for the student model. Figure 9 provides an example of this scenario. We coped with this limitation by introducing an algorithm selection procedure based on grid search optimization, where we select the most appropriate pair of models (teacher and student) for each dataset. To be clear, the teacher model is optimized for all approaches besides STUDD.

6.4 Experimental design choices

In the experiments, we evaluate different approaches according to predictive performance and labeling costs. This approach, which was taken before by dos Reis et al. (2016), directly quantifies the risk/reward of applying a specific detection mechanism. Notwithstanding, for completeness, we also measure traditional drift detection metrics, such as missed detection ratio or mean time to detection, using the approach SS as a benchmark.

Another important aspect of our experimental design is concept drift adaptation. We focus on typical batch learning classification models. After a change detection method launches an alarm it triggers an adaptation process in which a predictive model is retrained using a batch of recent observations. In principle, online learning systems can also be applied within our framework, e.g. the Adaptive Random Forest (Gomes et al., 2017). However, these methods typically self-adapt to the environment without explicit change detection method (Gama et al., 2014). Notwithstanding, as Gama et al. (2014) point out, explicit detection methods provide additional information regarding the dynamics of the relationship between the predictive model and the data.

In terms of application, we found that optimizing the predictive models (both teacher and student) is important. In terms of change detection the Page-Hinkley test provided interesting results, though other detection mechanisms are applicable.

7 Conclusions

Detecting concept drift is an important task in predictive analytics. Most of the state of the art approaches designed to tackle this problem are based on monitoring the loss of the underlying predictive model.

In this paper, we follow the idea that the assumption that labels are readily available for computing the loss of predictive models is too optimistic (Žliobaite, 2010; Pinto et al., 2019). Therefore, we focus on solving this problem in an unsupervised manner, i.e., without any access to the true labels.

We propose a method to deal with this task based on a model compression (Buciluă et al., 2006) approach. The core of the idea is to replace the loss of the predictive model which is deployed in the data stream (the teacher) with the *mimicking* loss of the student model as the input to traditional concept drift detection methods, such as the Page-Hinkley test (Page, 1954).

We carry out empirical experiments using 19 benchmark data streams and several state of the art methods. We show that the proposed method is able to detect concept drift and adapt itself to the environment. The behavior of the method is conservative with respect to other approaches, which is an advantage in domains where false alarms or label acquisition is costly. We published the code necessary to reproduce the experiments online. The data sets used are also available in public online repositories.

Author contributions Vitor Cerqueira, Heitor Murilo Gomes, Albert Bifet, Luis Torgo authors contributed to writing and research.

Funding The work of L. Torgo was undertaken, in part, thanks to funding from the Canada Research Chairs program.

Availability of data and materials All experiments and data are publicly available (c.f. footnote 1).

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Baena-Garcıa, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R., & Morales-Bueno, R. (2006). Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams* (Vol. 6, pp. 77–86).

Benavoli, A., Corani, G., Mangili, F., Zaffalon, M., & Ruggeri, F. (2014). A Bayesian Wilcoxon signed-rank test based on the Dirichlet process. In *International conference on machine learning* (pp. 1026–1034). PMLR.

Benavoli, A., Corani, G., Demšar, J., & Zaffalon, M. (2017). Time for a change: A tutorial for comparing multiple classifiers through Bayesian analysis. *The Journal of Machine Learning Research, 18*(1), 2653–2688.

Bifet, A. (2017). Classifier concept drift detection and the illusion of progress. In *International conference on artificial intelligence and soft computing* (pp. 715–725). Springer.

Bifet, A., & Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining* (pp. 443–448). SIAM.

Blackard, J. A., & Dean, D. J. (1999). Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture, 24*(3), 131–151.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Wadsworth International Group (Vol. 432, pp. 151–166).

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

Buciluǎ, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 535–541). ACM.

Caruana, R., Niculescu-Mizil, A., Crew, G., & Ksikes, A. (2004). Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on machine learning* (p. 18). ACM.

Cattral, R., Oppacher, F., & Deugo, D. (2002). Evolutionary data mining with automatic rule generalization. *Recent Advances in Computers, Computing and Communications, 1*(1), 296–300.

Cerqueira, V., Gomes, H. M., & Bifet, A. (2020). Unsupervised concept drift detection using a student–teacher approach. In *International conference on discovery science* (pp. 190–204). Springer.

Chang, C. C., & Lin, C. J. (2011). Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST), 2*(3), 1–27.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM sigkdd international conference on knowledge discovery and data mining* (pp. 785–794).

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement, 20*(1), 37–46.

de Souza, V. M., Silva, D. F., & Batista, G. E. (2013). Classification of data streams applied to insect recognition: Initial results. In *2013 Brazilian conference on intelligent systems* (pp. 76–81). IEEE.

Ditzler, G., & Polikar, R. (2012). Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge and Data Engineering, 25*(10), 2283–2301.

dos Reis, D. M., Flach, P., Matwin, S., & Batista, G. (2016). Fast unsupervised online drift detection using incremental Kolmogorov–Smirnov test. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1545–1554).

Dua, D., & Graff, C. (2017). Uci machine learning repository.

Fanaee-T, H., & Gama, J. (2014). Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence, 2*(2–3), 113–127.

Friedman, J., Hastie, T., & Tibshirani, R., et al. (2001). *The elements of statistical learning* (Vol. 1). Springer series in statistics, New York.

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. In *Brazilian symposium on artificial intelligence* (pp. 286–295). Springer.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR), 46*(4), 1–37.

Gao, J., Fan, W., Han, J., & Yu, P. S. (2007). A general framework for mining concept-drifting data streams with skewed distributions. In *Proceedings of the 2007 SIAM international conference on data mining* (pp. 3–14). SIAM.

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., et al. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning, 106*(9–10), 1469–1495.

Gomes, H. M., Read, J., Bifet, A., Barddal, J. P., & Gama, J. (2019). Machine learning for streaming data: State of the art, challenges, and opportunities. *ACM SIGKDD Explorations Newsletter, 21*(2), 6–22.

Gözüaçık, Ö., Büyükçakır, A., Bonab, H., & Can, F. (2019) Unsupervised concept drift detection with a discriminative classifier. In *Proceedings of the 28th ACM international conference on information and knowledge management* (pp. 2365–2368).

Hammami, N., & Bedda, M. (2010). Improved tree model for Arabic speech recognition. In *2010 3rd international conference on computer science and information technology* (Vol. 5, pp. 521–526). IEEE.

Harries, M., & Wales, N. S. (1999). Splice-2 comparative evaluation: Electricity pricing.

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Kaluža, B., Mirchevska, V., Dovgan, E., Luštrek, M., & Gams, M. (2010). An agent-based approach to care in independent living. In *International joint conference on ambient intelligence* (pp. 177–186). Springer.

Kim, Y., & Park, C. H. (2017). An efficient concept drift detection method for streaming data under limited labeling. *IEICE Transactions on Information and Systems, 100*(10), 2537–2546.

Kuncheva, L. I. (2004). Classifier ensembles for changing environments. In *International workshop on multiple classifier systems*, (pp. 1–15). Springer.

Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory, 37*(1), 145–151.

Losing, V., Hammer, B., & Wersing, H. (2015). Interactive online learning for obstacle classification on a mobile robot. In *2015 international joint conference on neural networks (ijcnn)* (pp. 1–8). IEEE.

Losing, V., Hammer, B., & Wersing, H. (2016). Knn classifier with self adjusting memory for heterogeneous concept drift. In *2016 IEEE 16th international conference on data mining (ICDM)* (pp. 291–300). IEEE.

Montiel, J., Read, J., Bifet, A., & Abdessalem, T. (2018). Scikit-multiflow: A multi-output streaming framework. *The Journal of Machine Learning Research, 19*(1), 2915–2914.

Page, E. S. (1954). Continuous inspection schemes. *Biometrika, 41*(1/2), 100–115.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research, 12,* 2825–2830.

Pinto, F., Sampaio, M. O., & Bizarro, P. (2019). Automatic model monitoring for data streams. arXiv preprint arXiv:1908.04240

Souza, V., Reis, D. M. D., Maletzke, A. G., & Batista, G. E. (2020). Challenges in benchmarking stream learning algorithms with real-world data. arXiv preprint arXiv:2005.00113.

Souza, V. M., Silva, D. F., Gama, J., & Batista, G. E. (2015). Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In *Proceedings of the 2015 SIAM international conference on data mining* (pp. 873–881). SIAM.

Vergara, A., Vembu, S., Ayhan, T., Ryan, M. A., Homer, M. L., & Huerta, R. (2012). Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical, 166,* 320–329.

Yu, S., Wang, X., & Principe, J.C. (2018). Request-and-reverify: Hierarchical hypothesis testing for concept drift detection with expensive labels. arXiv preprint arXiv:1806.10131.

Zhou, Z. H. (2018). A brief introduction to weakly supervised learning. *National Science Review, 5*(1), 44–53.

Zhu, X. (2010). Stream data mining repository. http://www.cse.fau.edu/xqzhu/stream.html.

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

Žliobaitė, I. (2011). Combining similarity in time and space for training set formation under concept drift. *Intelligent Data Analysis, 15*(4), 589–611.