Journal of the Brazilian Computer Society

RESEARCH Open Access

# Combining instance selection and self-training to improve data stream quantification

André G. Maletzke*, Denis M. dos Reis and Gustavo E. A. P. A. Batista

**Abstract**

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

**Keywords:** Data stream, Quantification, Concept drift

## Introduction

As machine learning becomes a mature discipline, researchers and practitioners start considering more challenging application settings. One example is learning from data streams, in which data arrive continuously and may be non-stationary. Recently, data streams have attracted a considerable amount of interest from the machine learning community. Consequently, we have seen an increasing and diverse set of methods in several tasks such as classification, clustering, and anomaly detection [1–4].

However, a relevant task, named quantification, has remained mostly unexplored in this setting. The objective in quantification is to provide an accurate estimation of the class distribution of an unlabeled set [5]. To elucidate the practical utility found in this task, consider as a concrete example the insect surveillance application described next.

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

Mechanical traps are the most used approach in insect surveillance. These traps attract and imprison insects, but require a human expert to manually identify and count the number of captured insects of interest. Such human-centered approach restricts the scale of application for insect surveillance. Recent research efforts have developed a sensor to automatically recognize the species of insects using data from the wing movement [6, 7]. This sensor uses machine learning algorithms to classify the signals obtained from the sensor into the insect species. For our purposes, classification is just an intermediate step. The final objective is quantification, i.e., counting the number of insects that belong to a particular species. These counts are the estimates of the local insect population.

*Correspondence: andregustavo@usp.br
Laboratório de Inteligência Computacional (LABIC), Instituto de Ciências Matemáticas e de Computação (ICMC), Universidade de São Paulo, São Carlos, Brazil

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

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

According to [5], quantification is a supervised task, recently formalized as a machine learning problem. Quantification shares similarities with classification. For instance, both consider the same representation for examples and a nominal output feature describing the class. However, quantification is not particularly interested in predicting the label for each instance, but in the overall quantity of elements of a specific class. Consequently, a quantifier issues an output for a set of examples, instead of for each instance. The output consists of a sequence of real values which are the estimates of the classes distribution.

Quantification and classification are also related tasks that benefit from each other. Quantification can be straightforwardly achieved by counting the number of examples classified as each class. This approach is known as *classify and count*. However, empirical evaluations, such as the one in [8], have shown that *classify and count* usually leads to suboptimal results. Including additional information, such as the estimated error and classification scores, leads to more accurate results. Nonetheless, even elaborated approaches that make use of additional information usually maintain an underlying classification nonetheless. On the other hand, classification benefits from quantification as an estimate of the class distribution is required to perform calibration [9].

To the best of our knowledge, the quantification research has mainly focused on the batch setting [5, 8, 10, 11]. In such cases, excepting changes in the proportions of the classes, other aspects of the distribution of data are assumed to be stationary. Recently, we proposed an algorithm to estimate the class distribution in non-stationary data streams [12]. This algorithm, named stream quantification by score inspection (SQSI), is resilient to changes in the feature space due to it monitoring the scores provided by a classifier and detecting concept drifts in an unsupervised manner. However, when SQSI detects a drift, it requests the true labels of all events in the chunk and updates the quantifier accordingly, acting as a supervised approach.

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

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

## Background

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

### Data stream

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

Real-world problems are increasingly modeled as data streams. This fact has motivated the development of novel methods that fulfill the requirements imposed by this type of data generation process, such as high volume, velocity, and volatility [4]. Also, mining data stream faces other challenges that include identifying and reacting to changes in the feature space and updating the classification model to incorporate those changes [13]. These changes in data, also known as concept drifts, are due to modifications in prior class probabilities $P(y_i)$, in data distributions $P(\vec{e}|y_i)$, and in class conditional probabilities $P(y_i|\vec{e})$. In general, two strategies are used to mitigate the concept drift impact: (*i*) adapt the models at regular intervals and (*ii*) use a trigger for concept drift presence and then adapt the model. There are several examples of methods that deal with concept drift in the literature [14, 15]. In our algorithm, we use the second approach and actively adapt a classification model when a drift is detected.

Another significant challenge in data stream mining is the occurrence of verification latency [14]. Most existing data stream classification algorithms assume that the true label of an instance becomes available as soon as these algorithms issue a classification. However, in real-world applications, this assumption rarely holds. Verification latency is the time delay before receiving the correct label [15]. Therefore, there usually is a delay of $T_l$ units of time until the actual label becomes available. Many methods assume $T_l = 0$, and consequently, these methods can update their models using labeled events more rapidly than possible in practice. Unfortunately, in real situations, $T_l \gg 0$, and sometimes $T_l = \infty$, which makes the data stream mining even more challenging. Verification latency remains little explored in data stream mining, and only in the last years, some methods have been proposed [2, 16].

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

## Quantification

Although quantification and classification share similarities, their objectives differ. Thus, quantification requires specific evaluation measures and, consequently, specialized machine learning algorithms [11].

Given the labeled set $E = \{\vec{e}_1, \ldots, \vec{e}_m\}$, where each event $\vec{e}_t$ has an associated label $y_t$, where $y_t \in C = \{c_1, c_2, \ldots, c_k\}$, we want to learn a classifier $\delta$ defined by the function $\delta : E \rightarrow C$ that assigns a label $c_i$ to each event $\vec{e}_t \in E$. The true frequency of a class $c_i$ in $E$ is defined by $freq_E(c_i) = \frac{|\{\vec{e}_t \in E\} | y_t = c_i|}{|E|}$, and the objective of a quantifier is to estimate $\widehat{freq}_E(c_i) \approx freq_E(c_i)$. These elements are sufficient to define the quantification problem [11] as stated in the Definition 1.

**Definition 1** (Quantification [11]) *Let $T_r$ be a labeled training set defined by collection E, an unlabeled test set $T_e$, and a classifier $\delta$ learned using $T_r$. The quantification problem consists in finding the best estimation of class distribution in $T_e$, i.e., $\forall c_i \in C$, we want to minimize the difference between the real frequency $freq_{T_e}(c_i)$ and the estimated one $\widehat{freq}_{T_e}(c_i)$.*

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

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

### *Classify and count (CC)* [8]

As previously mentioned, a straightforward approach to quantification is known as *classify and count* and consists in labeling each instance using a classifier and then counting the number of examples in each class. In other words, it estimates a class distribution by counting how many times each label was the output of a classifier $\delta$ for instances in a test set $T_e$, given that $\delta$ was induced with a training set $T_r$. We formalize this approach in Eq. 1.

$$\hat{p}^{CC}_{T_e} = p_{T_e}(\hat{c}_i) = \frac{|\{\vec{e}_t \in T_e | \delta(\vec{e}_t) = c_i\}|}{|T_e|} \tag{1}$$

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

In [8], the authors argue that this strategy is suboptimal since it does not take into account other pieces of information such as the marginal errors among the classes. For example, consider two binary classifiers: $h_1$ with $FP = 20$ and $FN = 20$, and $h_2$ with $FP = 18$ and $FN = 20$ ($FP$ and $FN$ are the number of false positives and false negatives, respectively). In this scenario, the classifier $h_2$ is better than $h_1$ in terms of classification. However, for quantification, the classifier $h_1$ is the best binary quantifier, since the $FP$ and $FN$ are equal, and the errors cancel out each other, resulting in a perfect quantifier. Summing up, according to Gao and Sebastiani [8], an inaccurate classifier can be an accurate quantifier if its marginal errors are spread as evenly as possible across FP and FN.

### *Probabilistic classify and count (PCC)* [8]

This approach computes $p_{T_e}(c_i)$ as an expected fraction of instances predicted to belong to class $c_i$. Let $p\left(c_i|\vec{e}_j\right)$ be the probability of $\vec{e}_j$ belonging to the class $c_i$, estimated by the classifier, and $E[\vec{e}_j]$ the expected value of $\vec{e}_j$ defined in Eq. 2.

$$\hat{p}^{PCC}_{T_e}(c_i) = E\left[p_{T_e}\left(\hat{c}_i\right)\right] = \frac{1}{|T_e|} \sum_{\vec{e}_t \in T_e} p\left(c_i|\vec{e}_t\right) \tag{2}$$

This method is a variation of CC and, in general, has shown good performance in comparison with other approaches [8].

***Adjusted classify and count (ACC)* [5]**

This method estimates the actual proportion of a given class by applying a correction based on true positive and false positive rates (*tpr* and *fpr*, respectively). Such rates are estimated with a validation set or with some cross-validation technique in $T_r$ (e.g., leave-one-out approach). Equation 3 presents the calculation of ACC for a given class $c_i$.

$$\hat{p}^{ACC}_{T_e}(c_i) = \frac{\hat{p}^{CC}_{T_e}(c_i) - fpr_{T_r}}{tpr_{T_r} - fpr_{T_r}} \tag{3}$$

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

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

## Evaluation metrics

Different measures can assess a quantifier accuracy. Most of them are based on error and cross-entropy measures [5]. Quantification also inherits virtually any measures that are suitable for regression problems. In this paper, we use an intuitive measure based on absolute error. Equation 4 defines the absolute error ($AE$). It corresponds to the average absolute difference between the class distribution estimation, $\hat{p}(c_i)$, and the actual class distribution, $p(c_i)$ [8].

$$AE\left(p, \hat{p}\right) = \frac{1}{|C|} \sum_{c_i \in C} \left|\hat{p}\left(c_i\right) - p\left(c_i\right)\right| \tag{4}$$

The $AE$ ranges between 0 (best) and $\frac{2(1-\min_{c_i \in C} p(c_i))}{|C|}$ (worst). Equation 5 presents the normalized absolute error ($NAE$), normalized version that ranges between 0 and 1 [8].

$$NAE\left(p, \hat{p}\right) = \frac{\sum_{c_i \in C} \left|\hat{p}\left(c_i\right) - p\left(c_i\right)\right|}{2\left(1 - \min_{c_i \in C} p\left(c_i\right)\right)} \tag{5}$$

However, according to Gao and Sebastiani [8] both $AE$ and $NAE$ may suffer from a serious issue when the true class prevalence is small. For instance, the same errors are produced for $p(c) = 0.01$ when predicting $\hat{p}(c) = 0.1$ and for $p(c) = 0.41$ when predicting $\hat{p}(c) = 0.50$. To avoid these cases, Gao and Sebastiani [8] propose the normalized relative absolute error ($NRAE$) as in Eq. 6.

$$NRAE\left(p, \hat{p}\right) = \frac{\sum_{c_i \in C} \frac{|\hat{p}(c_i) - p(c_i)|}{p(c_i)}}{|C| - 1 + \frac{1 - \min_{c_i \in C} p(c_i)}{\min_{c_i \in C} p(c_i)}} \tag{6}$$

$NRAE$ is undefined for zero denominators. Therefore, $p(c)$ and $\hat{p}(c)$ should be smoothed with Laplace smoothing, as in Eq. 7.

$$p_s(c) = \frac{\epsilon + p(c)}{\epsilon|C| + \sum_{c_i \in C} p(c_i)} \tag{7}$$

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

## Instance selection and semi-supervised learning

The original SQSI approach [12] splits the data stream into chunks with a non-overlapping sliding window and processes each chunk in order. Whenever SQSI reports a concept drift, it induces a new classifier that is trained with the data in the chunk, requesting true labels for the all of its instances.

Although SQSI has the merit of requesting labels only when it detects a concept drift, labels are frequently a limited and expensive resource. Therefore, in this paper, we propose the use of instance selection methods to find a subset of relevant examples in the chunk and request true labels only for them. We use a semi-supervised learning approach to expand these labels to the remaining examples in the chunk.

Instance selection methods are prevalent in the active learning literature. According to Settles [17], there are three basic settings:

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

In our work, since we process data streams in chunks, we can avail of pool-based approaches. According to Souza et al. [18], methods that select instances based on evaluation measures are widespread in the literature. These methods use approaches such as uncertainty, expected error reduction, and query committee to estimate the utility of each example. Uncertainty sampling selects instances that the classifier is least confident about their class labels. Expected error reduction selects examples that contribute to reducing the model error. Query by committee use ensembles to decide which examples to label, such as instances with the largest vote discordance.

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

#### *Random sampling*

Random sampling consists in selecting a subset of the instances randomly and with equal probability [19]. This technique can be considered naïve, since all instances are equiprobable and consequently no intelligent mechanism decides which ones are more valuable.

#### *Farthest-first traversal*

The intuition behind this method is that a representative labeled set is composed of a set of diverse events. To achieve such diversity, the farthest-first traversal algorithm selects $k$ instances that are the farthest from each other [18].

#### *Cluster-based sampling*

Clustering analysis has been used in several areas to identify groups and observe the relationship between the groups in an unsupervised manner [18, 19]. There are plenty of clustering methods, but, in essence, most of them employ the idea of similarity. Thus, the clustering objective consists in maximizing the similarity between the instances in the same group and in minimizing the similarity between examples from different groups. Therefore, a pool of events represented by a cluster structure can be useful to pick out representative events. Instance selection based on clustering algorithms selects the events near to cluster centers and near to the cluster borders, to compose a diverse set of events that will have their labels requested. Finding clusters in a pool of events usually involves a parameter $k$ that represents the number of clusters, and sometimes this parameter can be set according to the number of events that will be labeled. Another strategy consists in defining a number $b$ of events for which the true labels will be requested and a number $be$ of border events from each cluster. Hence, the number of clusters is given by $k = \lfloor b/(be+1) \rfloor$. Souza et al. [18] uses this last strategy.

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

Semi-supervised learning (SSL) methods are useful for learning tasks in which the quantity of labeled examples is so limited that compromises the generalization ability of the learning algorithm. Zhu and Goldberg [20] present a comprehensible survey of semi-supervised learning techniques. In our experimental section, we experiment with self-training, one of the most straightforward SSL methods available.

Self-training is a wrapper algorithm that iteratively applies a supervised learning method. This algorithm induces an initial classification model using the labeled portion of the data and uses this model to classify the unlabeled examples. In each iteration, part of the classified instances are moved to the labeled set. A popular approach is to move the instances classified with the highest confidence. The algorithm tags these instances with the predicted label. In our implementation, we decided to move one instance per iteration: the one with the highest score. The algorithm continues until it labels all examples. Algorithm 1 details the self-training strategy [18].

**Algorithm 1:** Self-training

```
  Input: Labeled instances $L_s$; Unlabeled instances $U_s$;
         Supervised machine learning method $ml_{alg}$
  Output: $\delta(U_s)$
1 begin
2     repeat
3         $\delta \leftarrow$ buildClassifier($L_s$, $ml_{alg}$)
4         $S \leftarrow$ SelectMostConfident($\delta$, $U_s$)
5         $U_s \leftarrow U_s - S$
6         $L_s \leftarrow L_s \cup S$
7     until $U_s = \emptyset$;
8     return $\delta(U_s)$
9 end
```

According to [20], self-training assumes that its predictions, when performed with high confidence, tend to be correct. This assumption has a controversial point motivated by the fact that early errors made by the classifier $\delta$ can be propagated to the entire unlabeled set. For this reason, we reinforce the importance of the instance selection methods to choose the most representative instances to be externally labeled by an oracle.

## Reference approaches for data stream quantification

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

### *Static*

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

### *Sliding*

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

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

## The original and extended SQSI algorithm

In this section, we present the SQSI algorithm, proposed by Maletzke et al. [12], as well as the extended version that incorporates instance selection and semi-supervised learning.

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

However, if the null hypothesis is rejected, we perform a li near transformation in the recent pool, applying Eq. 8 [15].

$$T\left(X_{f,i}^{act}\right) = \frac{X_{f,i}^{act} - mean\left(X_f^{act}\right)}{sd\left(X_f^{act}\right)} \times sd\left(X_f^{ref}\right) + mean\left(X_f^{ref}\right) \tag{8}$$

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

The intuition behind Eq. 8 is straightforward. The concept drift may be due to global linear transformation of the data. Therefore, SQSI first tries to reframe the attribute values in each example of the recent set with the mean and standard deviation of the reference set.

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

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

Figure 1 represents our proposed extension, named SQSI with instance selection (SQSI-IS). Similarly to the SQSI, our proposal requires an initial step that creates a quantifier with the first labeled chunk of the stream (training set). Thus, as result, we have our first quantifier $Q$. However, differently from SQSI, the SQSI-IS defines an instance selection method as well as the rate of the requested true labels.

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

In the aforementioned scheme, we present in Algorithm 2 our extension to the SQSI algorithm. The essence of the SQSI algorithm remains and the major difference between SQSI and SQSI-IS is in lines 15–18, where SQSI-IS requests a portion of the labels using an instance selection method and the self-training algorithm labels the remaining examples.

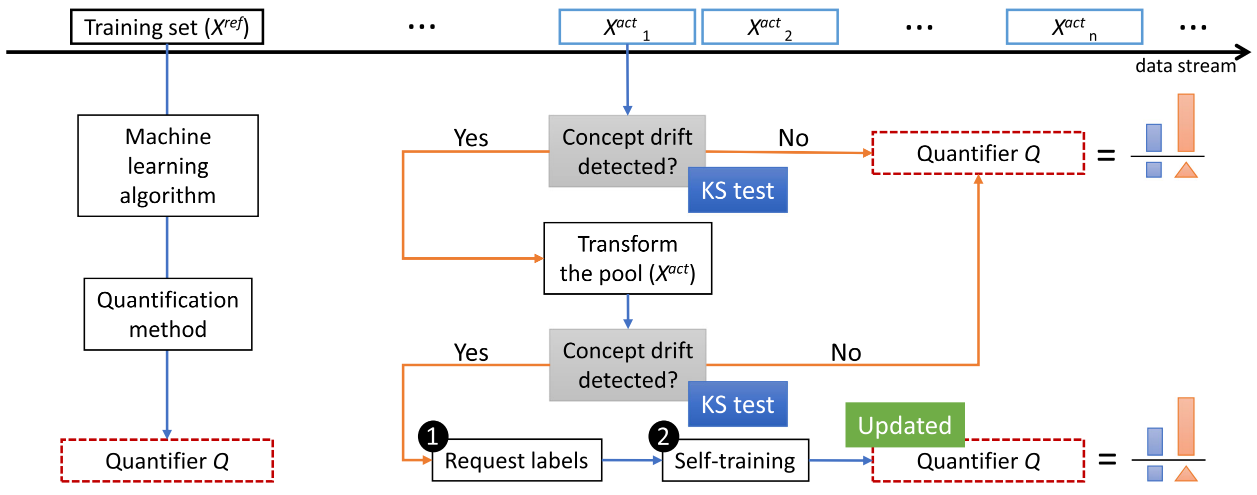

**Fig. 1** Schematic design of the SQSI-IS algorithm

**Algorithm 2:** SQSI with instance selection and semi-supervised learning

```
   Input: Initial training set $X^{ref}$; classifying algorithm
          $ml_{alg}$; data stream $E$; pool size $w$; quantification
          method $qnt_{method}$; instance selection method
          $IS_{method}$
   Output: Class prevalence for each pool $Qnt$
 1 begin
 2   $\delta \leftarrow$ buildClassifier($X^{ref}, ml_{alg}$)
 3   $X^{ref}_{sc} \leftarrow$ getScores($X^{ref}, ml_{alg}$)
 4   $X^{act} \leftarrow \emptyset$
 5   for each event $\vec{e} \in E$ do
 6     $X^{act} \leftarrow X^{act} \cup \{\vec{e}\}$
 7     if $|X^{act}| = w$ then
 8       $X^{act}_{sc} \leftarrow \delta(X^{act})$
 9       $t \leftarrow$ Test_KS($X^{ref}_{sc}, X^{act}_{sc}$)
10       if $t < 0.001$ then
11         $X^{act} \leftarrow$ transformData($X^{ref}, X^{act}$)
12         $X^{act}_{sc} \leftarrow \delta(X^{act})$
13         $t \leftarrow$ Test_KS($X^{ref}_{sc}, X^{act}_{sc}$)
14         if $t < 0.001$ then
15           $S_L \leftarrow$ requestLabels_withInstanceSelection($X^{act}$,
             $IS_{method}$)
16           $S_U \leftarrow X^{act} - S_L$
17           $X^{ref} \leftarrow$ Self-Training($S_L, S_U$,
             One-NearestNeighbor)
18           $\delta \leftarrow$ buildClassifier($X^{ref}, ml_{alg}$)
19         end
20       end
21       $Qnt \leftarrow Qnt \cup qnt\_method(\delta, X^{act})$
22       $X^{act} \leftarrow \emptyset$
23     end
24   end
25 end
26 return $Qnt$
```

## Time complexity analysis

The time complexity of SQSI-IS comprises four factors, as follows: (1) the complexity of performing the hypothesis test, (2) the complexity of applying the linear transformation exhibited in Eq. 8, (3) the complexity of the instance selection method, and (4) the complexity of training a new quantification model.

For the analysis, consider $w$ the size of each batch, $n$ the number of batches in the stream, $N$ the number of observations $n \times w$, $F$ the number of features, and $\gamma$ the proportion of batches that had true labels requested for.

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

For the second factor, the transformation exhibited in Eq. 8 requires only the mean and the standard deviation of both samples, which can be obtained linearly. Therefore, the time complexity of the transformation is $O(w)$ for a single batch and $O(N)$ for the whole stream.

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

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

## Experiments

Our experiments involve four real datasets that are described below:

- *Bike* [3] contains hourly records of a bicycle-sharing system with the corresponding weather and seasonal information between years 2011 and 2012. The goal is to predict whether there is high or low demand. Thus, we expect concept drift due to seasonality. It contains 17,379 instances;
- *Mosquitoes* has laboratory data with information of mosquitoes passing through a photosensitive sensor. The sensor data comprise seven features for each event: the wingbeat frequency (WBF) and the frequencies of the first six harmonics. The temperature varies during the stream, influencing the insect metabolism and, consequently, changing their wingbeat frequency [21, 22]. We consider the temperature as a latent variable for the quantification and classification tasks. The objective is to distinguish between events of female *Aedes aegypti* and *Aedes albopictus* mosquitoes from *Culex quinquefasciatus* and *Anopheles aquasalis*. This dataset contains 13,410 instances;
- *Insects* contains events generated by the same insect sensor. Differently from the Mosquitoes dataset, the features are the wingbeat frequency and the 92 first coefficients of the frequency spectrum obtained with a 1024-point fast Fourier transform. The object is to differentiate the *Aedes aegypti* mosquitoes from the insects *Musca domestica*, *Culex quinquefasciatus*, *Culex tarsalis*, and *Drosophila melonagaster*. It contains 83,339 instances;
- *NOAA* [2] is composed of meteorological conditions registered by the U.S. National Oceanic and Atmospheric Administration (Bellueve-Nebraska) for 50 years. This dataset contains eight features and 18,159 daily registers;
- *Arabic-Digit* [23]: A modified version of Arabic-Digit, described by a fixed number of MFCC values for the human speech of Arabic digits (among 10). The spoken digit defines the context, and the task is to predict the sex of the speaker. The dataset contains 26 features and 14,380 registers;
- *QG* [24] is a version of the dataset Handwritten [10], constrained to the handwritten letters *g*, and *q*. The context is defined by the author (among 10), and the objective is to predict the letter. This dataset is composed of 63 features and 13,279 registers.

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

To compare our results with [12], we retain the same experimental setup, i.e., we train an initial classifier with 400 examples, and the pool size has a fixed number of 300 examples. To analyze the impact of instance selection techniques, we evaluate the influence of different instance selection techniques in the SQSI-IS algorithm. We evaluate the following techniques: random sampling, farthest-first, and sampling based on clustering ($k$-means). We vary the portion of selected events from 5 to 95% by increments of 5%.

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

For cluster-based sampling, we use the $k$-means algorithm. We run the $k$-means algorithm to find two clusters ($k = \lfloor b/(be+1) \rfloor$). Thus, we request labels for the two events nearest to each cluster center and another eight true labels for the border events. In the experimental setup, we consider $be = 3$, motivated by results presented in [18].

Finally, after the labeling process, using one of the instance selection methods, we have two subsets of events. The first one has $b$% of labeled events and the second contains all remaining unlabeled examples. To avoid the risk of underfitting due to learning in an excessively small labeled sample, we apply the self-training algorithm that iteratively labels the remaining events, using a one-nearest neighbor classifier [20].

We execute all experiments ten times and report averaged results across the ten runs. We assess our results with NRAE and compare the methods using the Friedman test with 95% confidence level and Nemenyi post hoc test. Our evaluation is not a strictly prequential scheme, which means the tested examples are used for training only when a drift is detected. Additionally, for the SQSI-IS algorithm, the tested examples used for training are selected using an instance selection method. Implementation details and datasets are freely available as supplementary material [25].

## Results

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

We evaluated three different instance selection techniques: random, farthest-first, and clustering-based sampling (*k*-means). Our experiments are an extension of the results published in Maletzke et al. [12]. Therefore, we compare the results with the algorithms presented there, which include the *baseline*, the *topline* and the original version of the SQSI. We group the results by the combination of classification algorithm and quantification method.

Figures 2, 3, and 4 show the mean quantification errors for the random forest classifier applying distinct quantification methods and instance selection techniques for each dataset. The horizontal axis represents the percentage of examples labeled by the instance selection approach.

Figures 2, 3, and 4 show a decreasing error tendency, as the percentage of true labels increases. This decreasing tendency is common to several plots. In general, results with less than 50% labeled examples tend to have a higher variance than the ones labeled with more than 50% of the window.

Among the instance selection methods, clustering-based sampling performs best. Random and farthest-first techniques change places as second-best depending on the dataset and counting method. These results corroborate with the results presented in batch classification problems [17, 18].

SQSI performs well on datasets Bike, Insects, NOAA, Arabic-Digit, and QG. For these datasets, there is a clear separation between the *topline* and *baseline* quantifiers and SQSI tends to perform closer to the *topline*. For

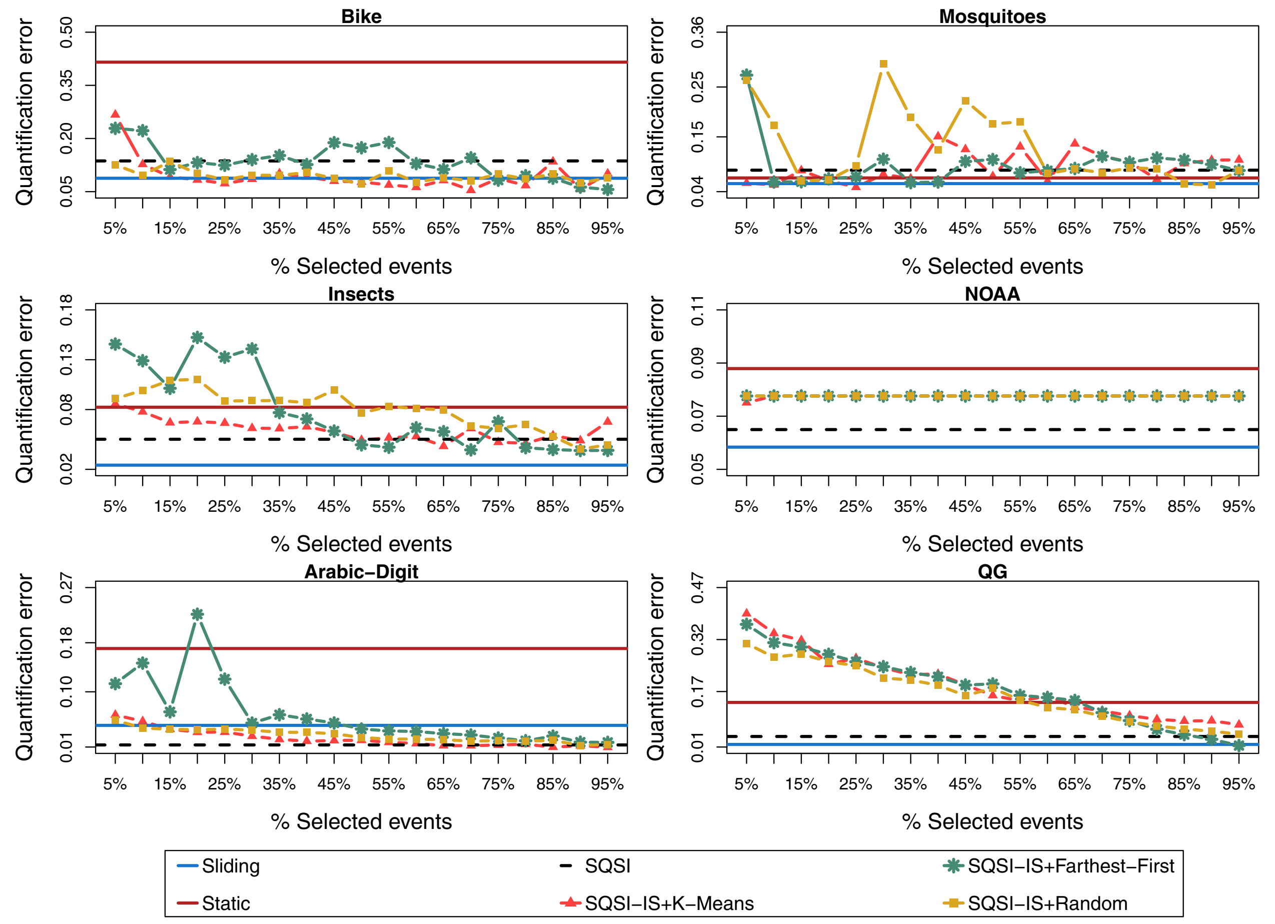

**Fig. 2** Mean quantification errors. Normalized relative quantification error for each dataset using the random forest (RF) as a classifier and the classify and count (CC) quantification method, varying the percentage of true labels requested by each instance selection technique

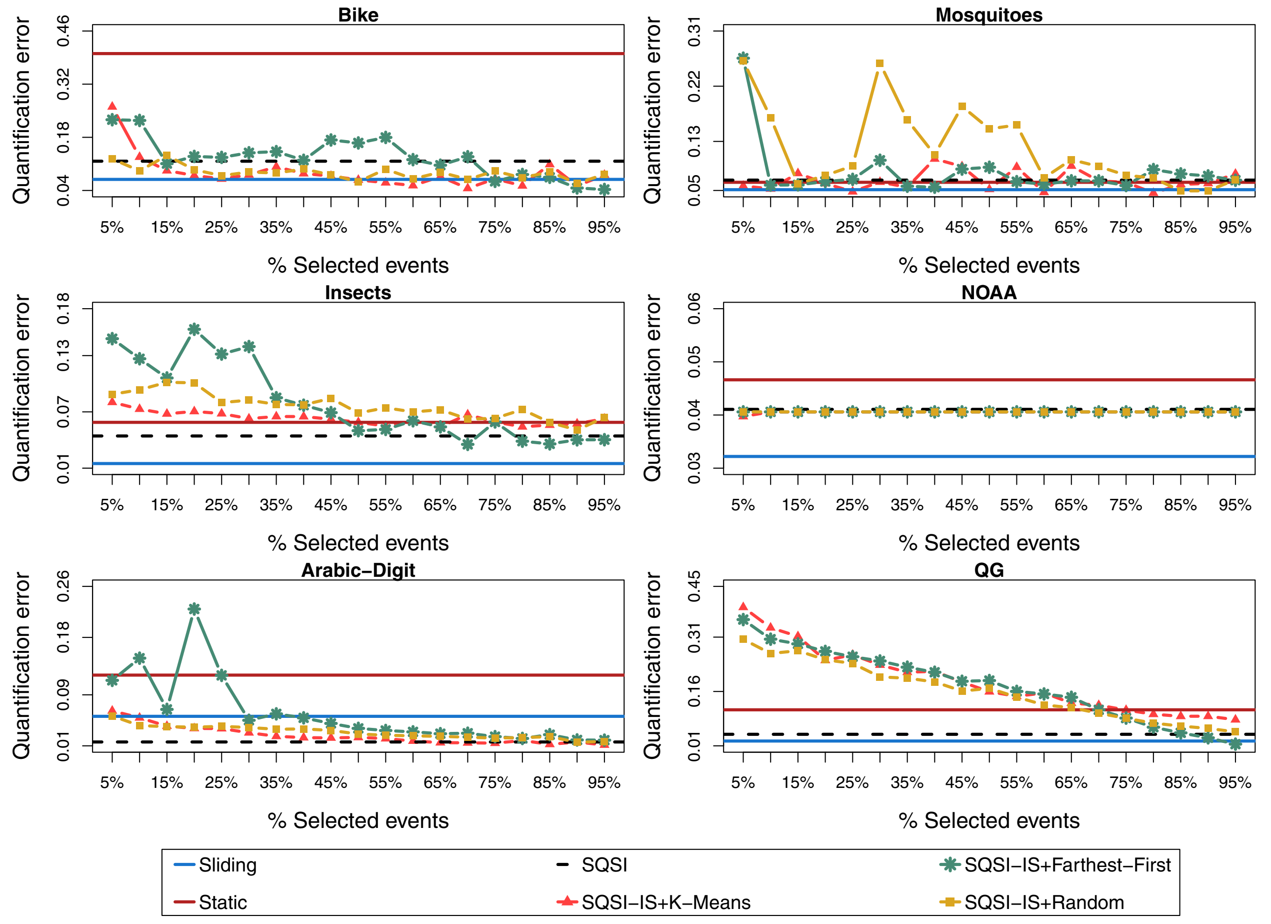

**Fig. 3** Mean quantification errors. Normalized relative quantification error for each dataset using the random forest (RF) as a classifier and the probabilistic classify and count (PCC) quantification method, varying the percentage of true labels requested in each active learning technique

Mosquitoes dataset, the performances of the *baseline* and *topline* are quite close to each other. This is an indication that no concept drifts are present or the occurrence of a significant amount of data chunks with a severely unbalanced class distribution. The unbalanced class distribution in the data chunks for each dataset will be discussed next. For the NOAA dataset, the KS test correctly identifies the lack of concept drifts. In total, KS flagged zero concept drifts with the RF classifier during the entire stream. Therefore, SQSI and SQSI-IS variants perform constantly across the entire stream.

For the Mosquitoes dataset, SQSI performs slightly worse than the baseline. However, in this case, the KS test flagged a concept drift in 18% of the pools. We hypothesize that most of these drifts are changes in the class distribution rather than in the feature space. We note that the KS statistic is sensitive to changes in both class and covariate attributes. These class changes are likely to be due to the circadian rhythm of the insects [6]. Additionally, a severe class distribution variability over some chunks of the streams has influenced the performance of the *topline* algorithm negatively. When a new chunk arrives, the *topline* algorithm makes a quantification and then receives the true labels for updating the quantification model. However, when the most recent chunk containing no drifts but a severely unbalanced class distribution, the *topline* efficiency goes down. Summing up, scenarios with no concept drift and with a severe class imbalance within the chunks are more promising to the *baseline* than the *topline*, because the latter will carry little information regarding one of the classes over to the following chunk of the data, while the former will keep its original, and possibly well balanced amount of information for each class. This rationale also applies to the SQSI and SQSI-IS algorithms, but for this a high drift detection rate is mandatory. Figure 5 shows histograms regarding the frequency of data chunks over different positive class distributions for each dataset.

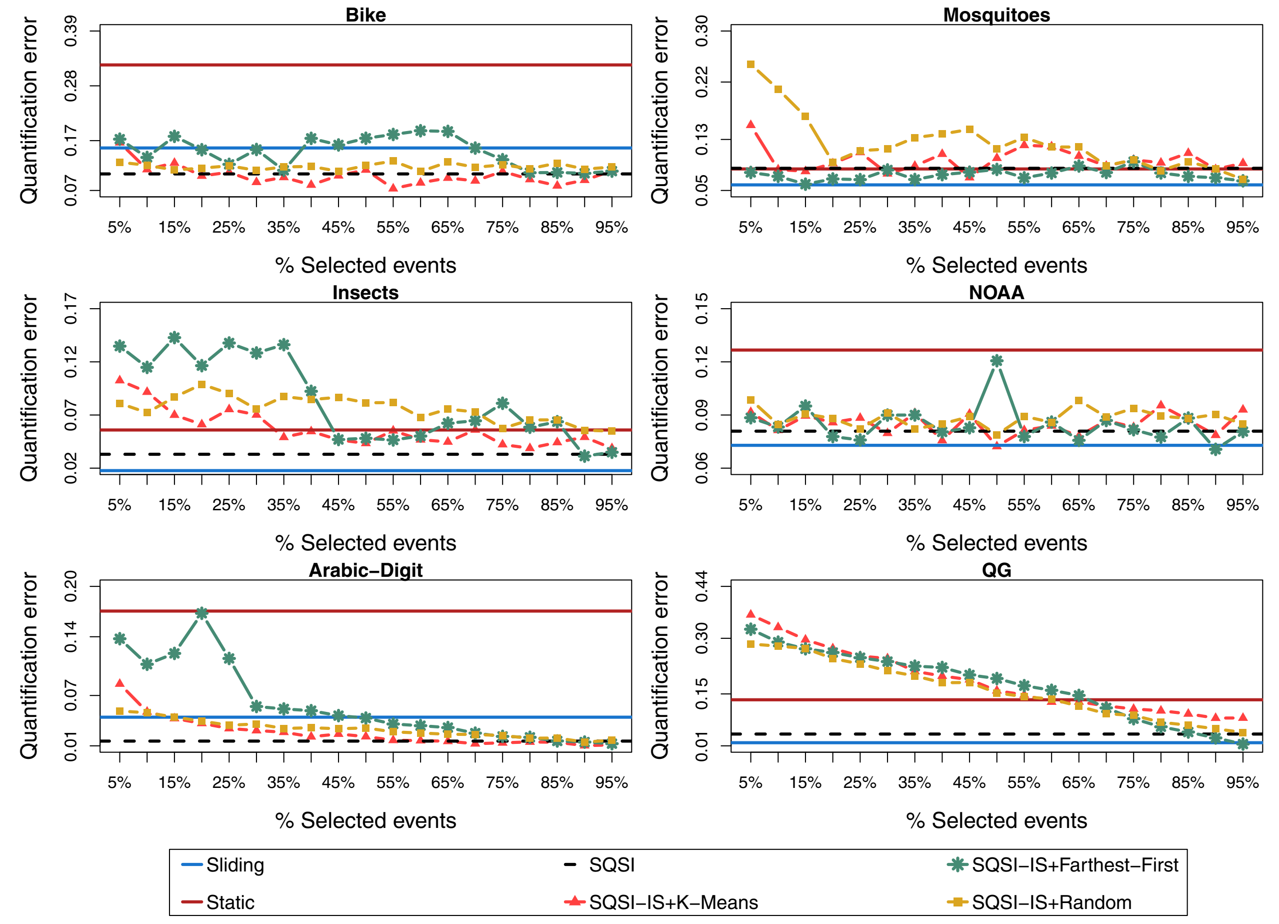

**Fig. 4** Mean quantification errors. Normalized relative quantification error for each dataset using the random forest (RF) as a classifier and the adjusted classify and count (ACC) quantification method, varying the percentage of true labels requested in each active learning technique

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

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

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

Regarding the quantification methods PCC performs slightly better than CC, and CC better than ACC. ACC relies on classifier error estimates to improve the quantification output. However, our experimental setup imposes two restrictions on ACC. The first one is that we train on a small samples and, therefore, it is more difficult to obtain reliable estimates. The second one is that, in the datasets with concept drifts, those estimates tend to become less reliable with time.

Summarizing the Figs. 2, 3, and 4, we show in Table 1 the mean NRAE about all datasets for each instance selection technique using different quantifiers applying RF as base classifier. Although the results showed lower errors when the *k*-means was used in conjunction with the PCC quantifier, the errors for all setups are quite similar.

To analyze these results in depth, we performed the Friedman test with Nemenyi post hoc analysis to verify whether the differences among instance selection techniques have statistical difference. For this, we applied a

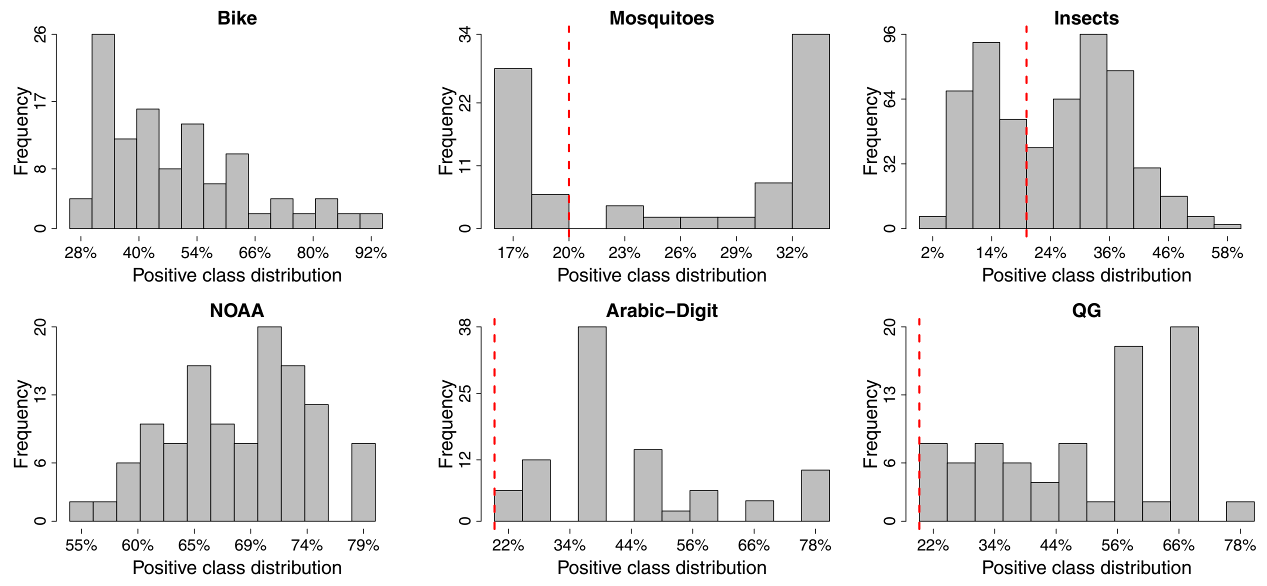

**Fig. 5** Positive class distributions histograms. Histograms regarding the frequency of data chunks over different positive class distributions for each dataset. The horizontal traced line represents 20% of the positive class distribution

comparison among $K$-Means, farther-first, and random techniques considering all range of positive class distributions, but separately for each quantification method. Table 2 presents the results, where $p$ value $>$ 0.05 is represented by ($ns$), $p$ value $\leq 0.05$ by ($*$), $p$ value $\leq 0.01$ by ($**$), and $p$ value $\leq 0.001$ by ($* * *$).

As expected, the statistical analysis shows that the clustering strategy (using $k$-means algorithm) outperforms, with statistically significance, the other strategies with the quantifiers PCC and ACC. Regarding the CC quantifier, the $k$-means strategy was statistically superior to the farthest-first. However, we did not find statistical differences between cluster-based sampling and random strategy with the CC quantifier.

Figures 6, 7, and 8 present the results for the SVM classifier.

The results for the SVM classifier corroborate with the results for random forrest. Once again, CC and PCC produced the best quantifiers and ACC presented slightly worse results. Once again, the *topline* has presented a bad performance for the Mosquitoes dataset that is affected by unbalanced data chunks as shown in Fig. 5.

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

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

In fact, the cluster-based sampling produces results with a lower variability compared to random and the farthest-first strategies. Furthermore, cluster-based sampling frequently produces results similar to (or better than) the original SQSI using less than 50% of the true labels. Summing up, considering our datasets and the experimental

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

| | CC | PCC | ACC | Mean |
|---|---|---|---|---|
| $k$-means | *0.0758* | *0.0670* | *0.0846* | *0.0758* |
| Farthest-first | 0.1053 | 0.0952 | 0.1102 | 0.1036 |
| Random | 0.1011 | 0.0867 | 0.0966 | 0.0948 |

For the sake of visualization, the best quantification errors are shown in italic

**Table 2** Statistical analysis

| | SQSI-IS+$k$-means | | | SQSI-IS+Farthest−first | | |
|---|---|---|---|---|---|---|
| | CC | PCC | ACC | CC | PCC | ACC |
| SQSI-IS+Farthest-first | $* * *$ | $* * *$ | $**$ | – | – | – |
| SQSI-IS+Random | *ns* | $*$ | $* * *$ | *ns* | *ns* | *ns* |

Results for Friedman test and Nemenyi post hoc test between the active learning techniques using RF classifier in each quantifier method
*ns* not significant

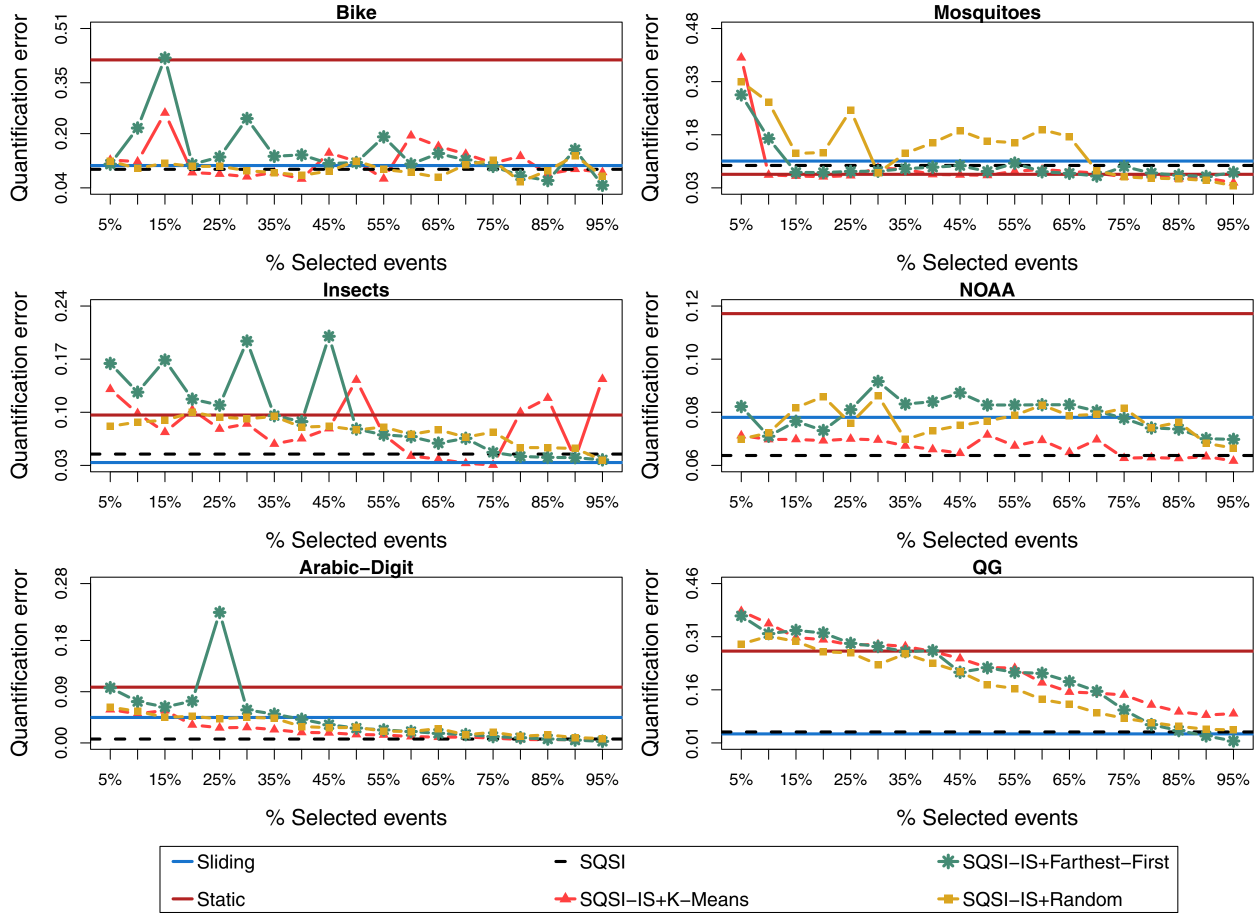

**Fig. 6** Mean quantification errors. Normalized relative quantification error for each dataset using the SVM as a classifier and the CC quantification method, varying the percentage of true labels requested in each active learning technique

setup, we point out the cluster-based sampling as the most promising instance selection technique among the evaluated ones. Indeed, instance selection methods produced equivalent or even better results using only a portion of the true labels. This remark is valid for both base classifiers.

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

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

In addition, when the quantifiers were induced using the SVM classifier, the differences between SQSI and SQSI-IS were greater. However, a statistical difference was not observed among all variations of SQSI-IS for all instance selection techniques. Adding a large number of datasets, the similarity between the SQSI and SQSI will be more evident. In general, our results suggest that achieving a confident quantification rate in scenarios with label scarcity is possible, especially when instance selection techniques are used.

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

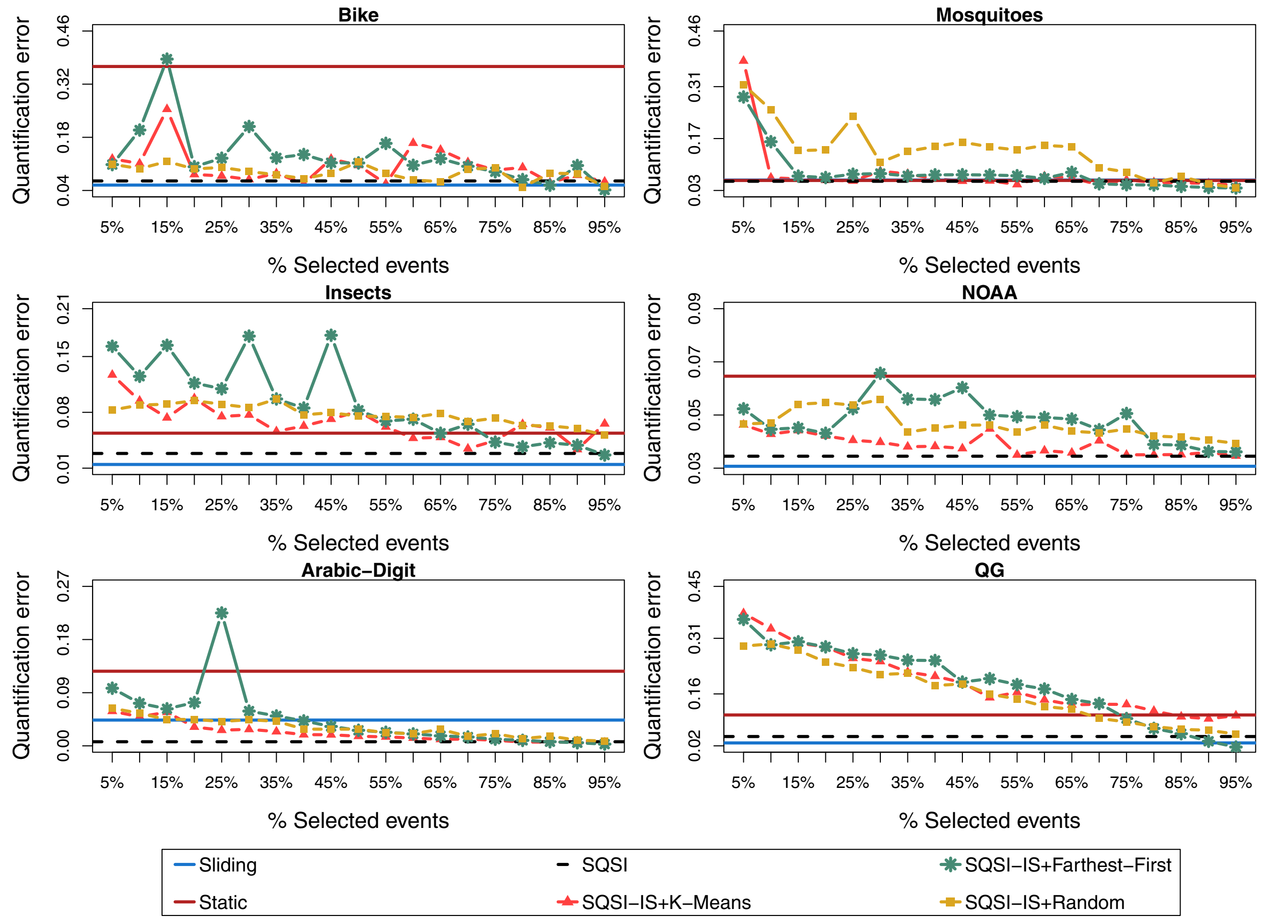

**Fig. 7** Mean quantification errors. Normalized relative quantification error for each dataset using the SVM as a classifier and the PCC quantification method, varying the percentage of true labels requested in each active learning technique

## Limitations

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

A second limitation is that we simplified the experiments by analyzing consecutive batches of data, instead of providing a quantification for the last $w$ instances after every single instance is observed. As $w$ is a small number compared with the size of the stream, this simplification should not significantly alter any performance measurement. Yet, this limitation can be withdrawn by using the Incremental Kolmogorov-Smirnov [15].

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

The fourth noteworthy limitation is that, as we feed the Kolmogorov-Smirnov test with the scores produced by the classifier, i.e, its confidence on each individual instance being positive, the test ends up being sensitive to changes in the proportion of the classes. This is counterproductive since, while changes in the prior distributions can negatively affect classifiers, this type of change without changes in the feature space should not affect the performance of quantifiers. Therefore, raising drift flags

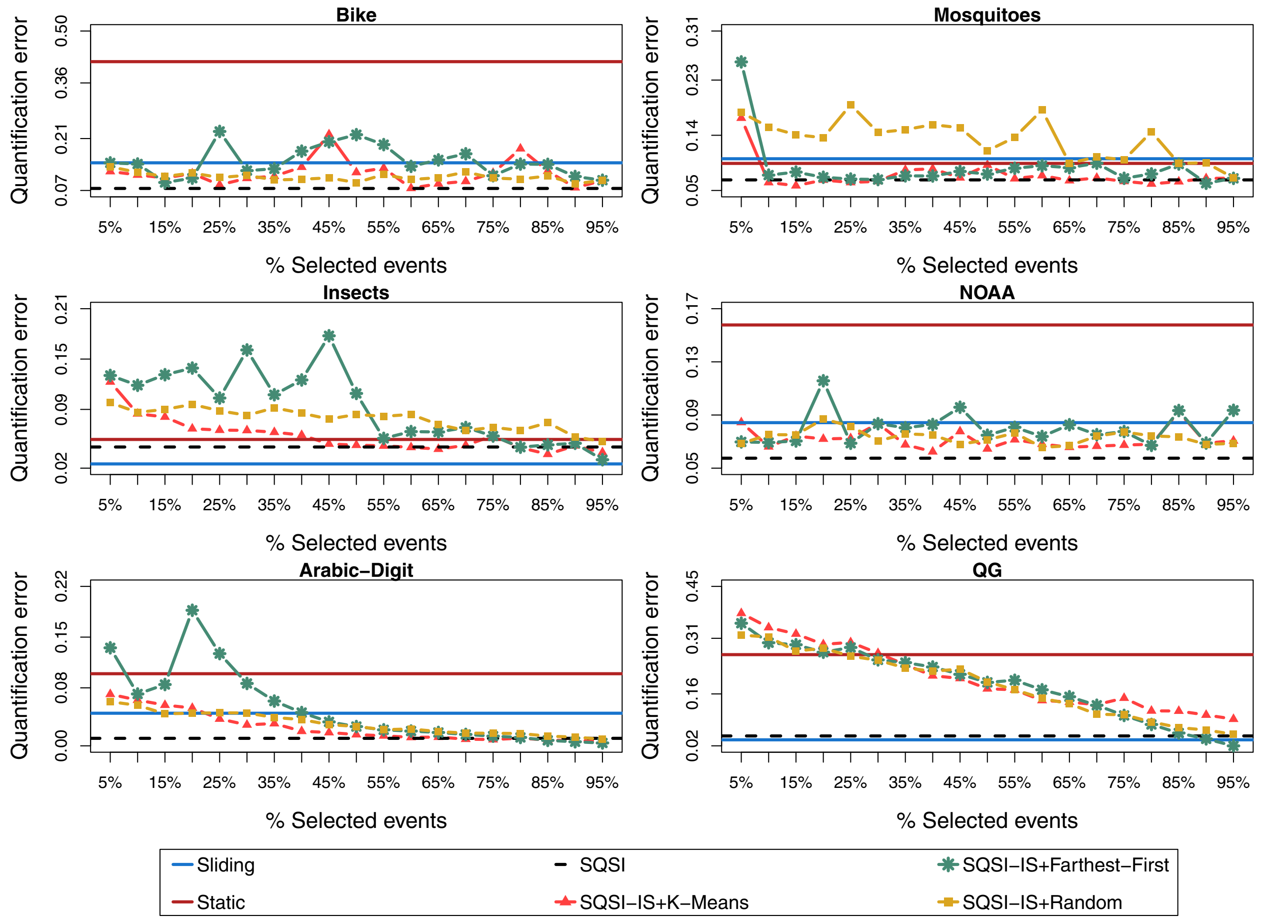

**Fig. 8** Mean quantification errors. Normalized relative quantification error for each dataset using the SVM as a classifier and the ACC quantification method, varying the percentage of true labels requested in each active learning technique

and requesting an updated model would be unnecessary for our quantification objective. For future work, we aim at replacing either what is used to feed the KS test or the test itself.

## Conclusions

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

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

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

| | CC | PCC | ACC | Mean |
|---|---|---|---|---|
| *k*-means | 0.1059 | *0.0758* | *0.0853* | *0.0890* |
| Farthest-first | *0.1000* | 0.0907 | 0.1164 | 0.1023 |
| Random | 0.1060 | 0.0953 | 0.0967 | 0.0993 |

For the sake of visualization, the best quantification errors are shown in italic

**Table 4** Statistical analysis

| | SQSI-IS+*k*-means | | | SQSI-IS+Farthest-first | | |
|---|---|---|---|---|---|---|
| | CC | PCC | ACC | CC | PCC | ACC |
| SQSI-IS+Farthest-first | * * * | * * * | * * * | – | – | – |
| SQSI-IS+Random | * | * * * | * * * | *ns* | *ns* | *ns* |

Results for Friedman test and Nemenyi post hoc test between the active learning techniques using SVM classifier in each quantifier method
*ns* not significant

**Table 5** Quantification error and statistical analysis

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

Mean error and results for Friedman test between the SQSI and SQSI-IS using different instance selection techniques for each quantification method
*ns* not significant

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

## Abbreviations
ACC: Adjusted classify and count; *AE*: Absolute error; CC: Classify and count; *fpr*: False positive rate; KS: Kolmogorov-Smirnov; *NAE*: Normalized absolute error; *NRAE*: Normalized relative absolute error; PCC: Probabilistic classify and count; RF: Random forest; SSL: Semi-supervised learning; SQSI: Stream quantification by score inspection; SQSI-IS: Stream quantification by score inspection with instance selection; SVM: Support vector machine; *tpr*: True positive rate; WBF: Wingbeat frequency

## Acknowledgements
The authors would like to thank the financial support from CAPES/Brazil, CNPq/Brazil, São Paulo Research Foundation (FAPESP)/Brazil, and United States Agency for International Development (USAID).

## Funding
This work was produced with grants CAPES (grant PROEX-6909543/D), CNPq (306631/2016-4), and FAPESP (grant 2016/04986-6). This material is based upon work supported by the United States Agency for International Development under Grant No AID-OAA-F-16-00072.

## Availability of data and materials
Please contact author for data requests.

## Authors' contributions
AGM and GEAPAB planned the SQSI's algorithm extension. AGM conducted the experimental evaluation. AGM and DMR analyzed the results and drafted the manuscript. All authors revised and approved the final manuscript.

## Competing interests
The authors declare that they have no competing interests.

## Publisher's Note

Received: 5 March 2018 Accepted: 22 August 2018
Published online: 12 October 2018

## References
1. Hill DJ, Minsker BS (2010) Anomaly detection in streaming environmental sensor data: a data-driven modeling approach. Environ Model Softw 25(9):1014–1022
2. Dyer KB, Capo R, Polikar R (2014) Compose: a semisupervised learning framework for initially labeled nonstationary streaming data. IEEE Trans Neural Netw Learn Syst 25(1):12–26. https://doi.org/10.1109/TNNLS.2013.2277712
3. Fanaee-T H, Gama J (2014) Event labeling combining ensemble detectors and background knowledge. Prog Artif Intell 2(2):113–127. https://doi.org/10.1007/s13748-013-0040-3
4. Nguyen HL, Woon YK, Ng WK (2015) A survey on data stream clustering and classification. KAIS 45:535–569. https://doi.org/10.1007/s10115-014-0808-1
5. Forman G (2005) Counting positives accurately despite inaccurate classification. In: ECML. Springer. pp 564–575
6. Chen Y, Why A, Batista GEAPA, Mafra-Neto A, Keogh E (2014) Flying insect classification with inexpensive sensors. J Insect Behav 27(5):657–677. https://doi.org/10.1007/s10905-014-9454-4
7. Silva DF, Souza VMA, Ellis D, Keogh E, Batista GEAPA (2015) Exploring low cost laser sensors to identify flying insect species. J Intell Robot Syst 80(1):313–330. https://doi.org/10.1007/s10846-014-0168-9
8. Gao W, Sebastiani F (2016) From classification to quantification in tweet sentiment analysis. Soc Netw Anal Min 6(1). https://doi.org/10.1007/s13278-016-0327-z
9. Vuk M, Curk T (2006) Roc curve, lift chart and calibration plot. Metodoloski zvezki 3(1):89
10. Bella A, Ferri C, Hernández-Orallo J, Ramírez-Quintana MJ (2010) Quantification via probability estimators. In: ICDM, Sidney. pp 737–742. https://doi.org/10.1109/ICDM.2010.75
11. Milli L, Monreale A, Rossetti G, Giannotti F, Pedreschi D, Sebastiani F (2013) Quantification trees. In: ICDM, Dallas. pp 528–536. https://doi.org/10.1109/ICDM.2013.122
12. Maletzke A, Reis D, Batista G (2017) Quantification in data streams: initial results. In: BRACIS, Uberlândia. pp 43–48. https://doi.org/10.1109/BRACIS.2017.74
13. Gama J, Medas P, Castillo G, Rodrigues P (2004) Learning with drift detection(Bazzan ALC, Labidi S, eds.). Springer, Berlin, Heidelberg
14. Masud M, Gao J, Khan L, Han J, Thuraisingham BM (2011) Classification and novel class detection in concept-drifting data streams under time constraints. IEEE Trans Knowl Data Eng 23(6):859–874. https://doi.org/10.1109/TKDE.2010.61
15. dos Reis DM, Flach P, Matwin S, Batista GEAPA (2016) Fast unsupervised online drift detection using incremental Kolmogorov-Smirnov test. In: ACM SIGKDD, San Francisco. pp 1545–1554. https://doi.org/10.1145/2939672.2939836
16. Souza VMA, Silva DF, Gama J, Batista GEAPA (2015) Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: SDM, Vancouver. pp 873–881. https://doi.org/10.1137/1.9781611974010.98
17. Settles B (2010) Active learning literature survey. Univ Wis Madison 52(55-66):11
18. Souza VMA, Rossi RG, Batista GEAPA, Rezende SO (2017) Unsupervised active learning techniques for labeling training sets: an experimental evaluation on sequential data. Intell Data Anal 21(5):1061–1095. https://doi.org/10.3233/IDA-163075
19. Zliobaite I, Bifet A, Pfahringer B, Holmes G (2014) Active learning with drifting streaming data. IEEE Trans Neural Netw Learn Syst 25(1):27–39
20. Zhu X, Goldberg AB (2009) Introduction to semi-supervised learning. Synth Lect Artif Intell Mach Learn 3(1):1–130
21. Taylor L (1963) Analysis of the effect of temperature on insects in flight. J Anim Ecol 32(1):99–117
22. Mellanby K (1936) Humidity and insect metabolism. Nature 138:124–125

23. Hammami N, Bedda M (2010) Improved tree model for arabic speech recognition. In: ICCSIT. IEEE, Chengdu Vol. 5. pp 521–526
24. dos Reis D, Maletzke A, Batista G (2018) Unsupervised context switch for classification tasks on data streams with recurrent concepts. In: ACM/SIGAPP. ACM, Pau, France Vol. 33
25. Maletzke A, dos Reis D, Batista G (2018) Combining instance selection and self-training to improve data stream quantification, Online Supplementary Material. https://sites.google.com/site/andregustavom/research/sqsi-is. Accessed 04 June 2018