Machine Learning (2020) 109:175–218
https://doi.org/10.1007/s10994-019-05840-z

Kappa Updated Ensemble for drifting data stream mining

Alberto Cano[1] · **Bartosz Krawczyk**[1]

Received: 13 November 2018 / Revised: 29 June 2019 / Accepted: 6 September 2019 /
Published online: 2 October 2019

Abstract
Learning from data streams in the presence of concept drift is among the biggest challenges of contemporary machine learning. Algorithms designed for such scenarios must take into an account the potentially unbounded size of data, its constantly changing nature, and the requirement for real-time processing. Ensemble approaches for data stream mining have gained significant popularity, due to their high predictive capabilities and effective mechanisms for alleviating concept drift. In this paper, we propose a new ensemble method named Kappa Updated Ensemble (KUE). It is a combination of online and block-based ensemble approaches that uses Kappa statistic for dynamic weighting and selection of base classifiers. In order to achieve a higher diversity among base learners, each of them is trained using a different subset of features and updated with new instances with given probability following a Poisson distribution. Furthermore, we update the ensemble with new classifiers only when they contribute positively to the improvement of the quality of the ensemble. Finally, each base classifier in KUE is capable of abstaining itself for taking a part in voting, thus increasing the overall robustness of KUE. An extensive experimental study shows that KUE is capable of outperforming state-of-the-art ensembles on standard and imbalanced drifting data streams while having a low computational complexity. Moreover, we analyze the use of Kappa versus accuracy to drive the criterion to select and update the classifiers, the contribution of the abstaining mechanism, the contribution of the diversification of classifiers, and the contribution of the hybrid architecture to update the classifiers in an online manner.

Keywords Machine learning · Data streams · Concept drift · Classification · Ensemble learning

1 Introduction

The data revolution over the last two decades has changed almost every aspect of data analytics. One must take into account the fact that the size of data is constantly growing and

Editor: João Gama.

✉ Alberto Cano
acano@vcu.edu

Bartosz Krawczyk
bkrawczyk@vcu.edu

1 Virginia Commonwealth University, 401 W. Main St. E4251, Richmond, VA 23284, USA

one cannot store all of it. Data is in motion, constantly expanding, and changing its properties (Morales et al. 2016). Additionally, data may come from many sources at the same time, calling for efficient preprocessing and standardization (Ramirez-Gallego et al. 2017). Such changes affected various real-life applications, including social media (Miller et al. 2014), medicine (Triantafyllopoulos et al. 2016), and security (Faisal et al. 2015) to name a few. This poses challenges for learning systems that must accommodate all these properties, while maintaining a high predictive power and capabilities of operating in real-time (Marrón et al. 2017; Ramírez-Gallego et al. 2017; Cano and Krawczyk 2018, 2019).

The velocity of data gave rise to the notion of data streams, potentially unbounded collections of data that continuously flood the system. As new data is continuously arriving, storing a data stream is not a viable option. One needs to analyze new instances on-the-fly, incorporate the useful information into the classifier, and discard them. Both the prediction and classifier update steps cannot be of a high complexity, as instances arrive rapidly and bottlenecks must be avoided. Data streams are also subject to a phenomenon known as concept drift (Gama et al. 2014; Žliobaite et al. 2015a; Barddal et al. 2017), where the properties of the stream are subject to a change over time. This includes not only the discriminatory power of features but also the size of the feature space, ratios of instances among classes, as well as the emergence and disappearance of classes.

In order to accommodate such characteristics, data streams inspired the development of new families of algorithms capable of continuous integration of new data, while providing robustness to its evolving nature. These include concept drift detectors for creating alarms when the change takes place, and incremental or online algorithms that are capable of processing new instances as they arrive and discarding them right after (Pears et al. 2014). Ensemble techniques, one of the most promising directions in standard machine learning, have been successfully applied to the both former (Woźniak et al. 2016) and the latter domains (Krawczyk et al. 2017). In this paper, we will focus on data streams classification. Although there is a plethora of existing ensemble methods for drifting data streams, none of the existing algorithms offers a stable performance over a high variety of potential streaming problems (Krawczyk et al. 2017; Gomes et al. 2017a). This calls for the development of new ensemble classifiers that will be characterized by a lower variance in the results when subject to varying types of complex data.

Furthermore, existing ensemble techniques are dedicated to either standard or imbalanced datasets but not to both of them (Hoens et al. 2012; Ren et al. 2018b). Standard ensembles fail when dealing with skewed data (Krawczyk 2016), while ensembles dedicated to imbalanced data streams perform sub-par to standard methods when the number of instances among classes is roughly equal (Krawczyk et al. 2017). Additionally, one may not know beforehand that the analyzed data stream is imbalanced, or data stream may become periodically imbalanced due to variations in ratios of arriving instances. This calls for developing a more robust approach that can work efficiently in both scenarios.

This paper proposes a new ensemble learning algorithm for drifting data streams, named Kappa Updated Ensemble (KUE). It addresses the discussed shortcomings of existing ensemble methods, offering a stable and efficient classification approach over a wide set of data stream problems. Additionally, it displays an improved robustness to class imbalance without any need for applying preprocessing or specialized base classifiers. KUE achieves this by guiding its learning process using the Kappa statistic and utilizing it for the calculation of weights assigned to base classifiers. KUE offers a combination of online and block-based approaches, both continuously updating its base classifiers and replacing them with new ones when necessary. What is important, KUE adds new classifiers only when they improve the ensemble performance, while maintaining the previous learners in opposite cases. Base classifiers used by KUE are diversified by using random feature subsets and updated with new instances with probability derived from Poisson distribution. The wider exploration of feature subspaces leads to improved generalization capabilities and better anticipation of potential concept drifts. Finally, base classifiers in KUE may abstain from voting, reducing the chance of incompetent classifiers affecting the final decision. Despite its various characteristics, KUE displays low decision and update times, as well as low computational resources consumption.

To summarize, the main contributions of this work are as follow:

- KUE, a new ensemble classification algorithm for drifting data streams that uses the Kappa statistic for selecting and weighting its base classifiers.
- Hybrid architecture, that updates base classifiers in an online manner, while changing the ensemble set-up in a block-based mode.
- Diversification techniques for base learners that combine online bagging with using random feature subspaces.
- Abstaining of base classifiers that reduces the impact of non-competent base learners.
- Achieving stable performance over a variety of streaming problems, while maintaining robustness to drift and class imbalance, and displaying low computational complexity.
- A thorough experimental study, comparing KUE to 15 state-of-the-art ensemble methods over 60 standard and 33 imbalanced data stream benchmarks.
- An analysis of the contribution and impact of each of the algorithm's mechanisms individually.

The rest of the paper is organized as follows. Section 2 presents an overview of data stream mining and related works in ensemble learning and imbalanced classification for data streams. Section 3 provides a detailed description of the proposed Kappa Updated Ensemble algorithm, its architecture, and principles. Section 4 presents a thorough experimental study on a large set of data streams, including imbalanced streams with concept drift and varying imbalance ratio. Moreover, the contribution of each of the algorithm's mechanisms is individually analyzed to evaluate their impact in the quality of the predictions. Experimental results are also validated through non-parametric statistical analysis. Finally, Sect. 5 summarizes the concluding remarks and discusses future lines of work.

2 Data stream mining

This section presents a comprehensible overview of data stream mining, concept drift, ensemble classifiers for data streams, and introduces the challenge of imbalanced learning in data stream mining.

2.1 Overview

A data stream can be seen as a sequence $< S_1, S_2, \ldots, S_n, \ldots >$, in which each element S_j is a set of instances (or a single instance in a case of online learning) (Gaber 2012). Each instance is independent and randomly generated using a stationary probability distribution D_j. In this paper, we consider the supervised learning scenario that allows us to define each element as:

$$S_j \sim p_j(x^1, \ldots, x^d, y) = p_j(\mathbf{x}, y), \quad (1)$$

where $p_j(\mathbf{x}, y)$ is a joint distribution of jth instance, defined by d-dimensional feature space and belonging to class y. Each instance in the stream is independent and randomly drawn from a stationary probability distribution $\Psi_j(\mathbf{x}, y)$.

If a transition $\mathrm{S}_j \rightarrow \mathrm{S}_{j+1}$ (where $\mathrm{D}_j = \mathrm{D}_{j+1}$) holds, then we deal with a stationary data stream. However, real-life problems are usually subject to a change over time, where the characteristics and definitions of a stream evolve. This phenomenon is known as concept drift (Brzeziński and Stefanowski 2013; Gama et al. 2014; Balle et al. 2014; Webb et al. 2016, 2018).

We will now present main aspects related to concept drift and other difficulties present in evolving data streams:

- *Influence on decision boundaries* There is a distinction between real and virtual concept drifts (Sobolewski and Woźniak 2013). The former influences previously learned decision rules or classification boundaries, increasing the error on instances coming from the current stream concept. Real drift affects posterior probabilities $p_j(y|\mathbf{x})$ and additionally may impact unconditional probability density functions. It poses a significant threat to the learning system and must be tackled as soon as it appears. Virtual concept drift affects only the distribution of features $\mathbf{x}$ over time:

$$\widehat{p}_j(\mathbf{x}) = \sum_{y \in Y} p_j(\mathbf{x}, y), \tag{2}$$

  where Y is a set of possible values taken by S_j. As only the values of features change, this type of drift does not force us to adapt the used classification model. However, it may trigger false change alarms and thus force unnecessary and costly adaptations.
- *Locality of changes* We distinguish between global and local concept drifts (Gama and Castillo 2006). The former one affects the entire stream, while the latter one affects only certain parts of it (e.g., selected regions of the feature space or subsets of classes). These types of drifts should be distinguished, as often rebuilding the entire classification model is not necessary and one may concentrate in updating only the part of the learning system that has been subject to a local concept drift.
- *Speed of changes* We distinguish between sudden, gradual, and incremental concept drifts (Gama et al. 2014).
  Sudden concept drift describes a scenario in which underlying instance distribution abruptly changes with tth example arriving from the stream:

$$p_j(\mathbf{x}, y) = \begin{cases} D_0(\mathbf{x}, y), & \text{if } j < t \\ D_1(\mathbf{x}, y), & \text{if } j \geq t \end{cases} \tag{3}$$

  Incremental concept drift can be seen as a steady progression from one concept to another (thus consisting on multiple intermediate concepts in between), such that the distance from the old concept is increasing, while the distance to the new concept is increasing:

$$p_j(\mathbf{x}, y) = \begin{cases} D_0(\mathbf{x}, y), & \text{if } j < t_1 \\ (1-\alpha_j)D_0(\mathbf{x}, y) + \alpha_j D_1(\mathbf{x}, y), & \text{if } t_1 \leq j < t_2 \\ D_1(\mathbf{x}, y), & \text{if } t_2 \leq j \end{cases} \tag{4}$$

  where

$$\alpha_j = \frac{j - t_1}{t_2 - t_1}. \tag{5}$$

Gradual concept drift stands for a situation where over a given period of time instances arriving from the stream oscillate between two distributions:

$$p_j(\mathbf{x}, y) = \begin{cases} D_0(\mathbf{x}, y), & \text{if } j < t_1 \\ D_0(\mathbf{x}, y), & \text{if } t_1 \leq j < t_2 \wedge \delta > \alpha_j \\ D_1(\mathbf{x}, y), & \text{if } t_1 \leq j < t_2 \wedge \delta \leq \alpha_j \\ D_1(\mathbf{x}, y), & \text{if } t_2 \leq j, \end{cases} \tag{6}$$

where $\delta \in [0, 1]$ is a random variable. This models the decreasing probability of old concept occurrence with increasing probability of the new concept occurrence.

- *Recurrence* In many scenarios it is possible that a previously seen concept from kth iteration may reappear $\mathrm{D}_{j+1} = \mathrm{D}_{j-k}$, once or periodically (Gama and Kosina 2014). This is known as recurring concept drift.
- *Presence of noise* Apart from concept drift, one may encounter other types of changes in data. They are connected with the potential appearance of incorrect information in the stream, and known as blips and noise (Zhu et al. 2008; Chandola et al. 2009). Blips are random changes in stream characteristics that should be ignored (may be seen as outliers). Noise represents significant fluctuations in feature values or class labels, representing some corruption in received instances. While the classification model should adapt to concept drift, these types of changes should not influence the underlying model, as they will have a negative impact.
- *Feature drift* This is a type of change in data streams that happens when a subset of features becomes, or stops to be, relevant to the learning task (Barddal et al. 2017). Additionally, new features may emerge (thus extending the feature space), while the old ones may cease to arrive (Barddal et al. 2019a). Therefore, classifiers need to adapt to these changes in feature space (Barddal et al. 2016) by performing a dynamic feature selection (Yuan et al. 2018; Barddal et al. 2019b), using randomness in selected features (Abdulsalam et al. 2011), or employing a sliding window and feature space transformation (Nguyen et al. 2012).

We must note that in most real-world problems the nature of changes is far from being well-defined or known, and we must be able to deal with hybrid changes through time, known as mixed concept drift. Moreover, this becomes even more challenging when access to the labels is unavailable (Sethi and Kantardzic 2017).

The simplest solution for handling concept drift is to rebuild the classification model whenever new data becomes available. Such an approach has a prohibitive computational cost and it is not feasible for any real-life applications (Žliobaite et al. 2015b; Matuszyk and Spiliopoulou 2017; Srinivasan and Bain 2017). This has led to the development of specialized methods for this problem. There are two main approaches for tackling concept drift:

- *Explicit drift handling* This approach is based on using an external tool, called detector, that monitors specific characteristics of a data stream (Kuncheva 2013; Barros and Santos 2018). Most typical ones include changes in classification errors (Pesaranghader and Viktor 2016), statistical distribution variations (Sobolewski and Woźniak 2017), or density changes (Liu et al. 2018). Detectors output two types of information: warning and detection. A warning signal is being raised when a start of potential changes is being observed and informs the learning system to start training a new classifier on recent instances. A detection signal is being emitted when the magnitude of changes reaches a certain threshold and informs the learning system to replace the old classifier with a new one.

- *Implicit drift handling* Here, we assume that the used classification model inherently adapts itself to changes. One of the earliest approaches was to use a sliding window of fixed size that stores most recent instances from the stream (Zhang et al. 2017). By incorporating new instances and discarding old ones, this solution achieves an adaptation to drifts in streams. A problem of proper window size setting is predominant here. Too small window will adapt swiftly even to small changes but may lead to overfitting on a small training sample. A large window will capture a more global outlook on the stream but may mix instances coming from different concepts. Recent works in this area propose to use multiple windows or to dynamically adapt the window size (Mimran and Even 2014). Another approach lies in using online learners, capable of processing instances from the stream one by one, thus focusing its learning procedure on the most recent ones (Vicente et al. 1998). Online learners are characterized by high processing speed and low computational complexity and must process each instance only once. Some standard classifiers (i.e., neural networks) are capable of working in an online mode but there is a plethora of specialized classifiers that use modified learning schemes to cope with drift presence (Zhang et al. 2016).

Please note that only explicit methods actually detect the moment of drift, are capable of pinpointing the moment of change, and acting accordingly. They return information about the change and thus can be seen as actual detection. Implicit methods follow the data and inherently adapt to changes, but they (in vast majority of cases) do not return any information on when the change took place, what was the nature of change, etc. Therefore, they cannot be considered as "detection" methods. Thus, we refer to them both as explicit and implicit drift handling methods.

2.2 Ensemble learning for data stream mining

Ensemble learning has gained significant attention in machine learning and data mining over the last two decades. Combining multiple classifiers is capable of returning an improved predictive performance over any single learner in the pool. For an ensemble to work, it must be formed from mutually complementary and individually competent classifiers. This is the problem of diversity—a combination of accurate, yet similar learners will contribute no new knowledge, while a combination of different, yet inaccurate classifiers will create a weak ensemble. Therefore, various techniques for controlled diversity creation has been proposed, with the most popular ones being Bagging, Boosting, and Random Forest (Bertini and Nicoletti 2019; Van Rijn et al. 2018; Bertini and Nicoletti 2019).

Ensemble approaches are very popular in data stream mining, which can be contributed to their flexible set-up, capabilities of changing the importance of base classifiers, as well as natural mechanisms for incorporating new information (Krawczyk et al. 2017; Gomes et al. 2017a; Dong and Japkowicz 2018). Additionally, new incoming data can be seen as an attractive way to maintain diversity among ensemble members (Minku et al. 2010). We may distinguish three main approaches for learning ensemble classifiers over data streams:

- *Weight modifications* This approach focuses on modifying the weights assigned to classifiers in the ensemble, in order to reflect their current competencies over the data stream (Kolter and Maloof 2007; Ren et al. 2018a). The basic idea lies in having a diverse pool of classifiers and monitoring their performance in a dynamic way (e.g., instance by instance). Classifiers that make correct predictions can be deemed as better adapted to the current concept and thus their weights should be increased. Classifiers making incorrect decisions are penalized in a similar manner (Mejri et al. 2018). More advanced solutions

take into an account the presence of concept drift that should strongly affect the weights, especially in case of sudden changes (Krawczyk et al. 2017). The weight adaptation speed after a concept drift must be much more rapid to reflect the new state of the environment.
- *Dynamic ensemble line-up* This approach focuses on dynamically replacing the classifiers in the pool. After a new chunk of data becomes available, a new classifier is being trained on it and added to the ensemble (Brzeziński and Stefanowski 2014b). If a certain ensemble size has been reached, a pruning mechanism is applied to remove irrelevant learners (Ditzler et al. 2013). Usually, the oldest or weakest performing classifier is being discarded. This mechanism implements both incremental learning of new concepts, as well as gradual forgetting of old ones, thus naturally tackling the evolving nature of data streams. Dynamic ensemble set-up is usually connected with specific weighting mechanisms that promote the newest ensemble members and reduce the weights as time passes (Jackowski 2014). Recent proposals postulate to boost the weights of classifiers if they are performing well on current instances even if these learners are trained on older concepts (Woźniak et al. 2013).
- *Online ensemble update* This approach focuses on maintaining a pool of online classifiers that are updated with incoming instances (Pietruczuk et al. 2017; Zhai et al. 2017; Pesaranghader et al. 2018). Here the set-up is stable and learners adapt to drifts by updating them with new data (Olorunnimbe et al. 2018; Bonab and Can 2018). Additionally, this is used to maintain diversity among base classifiers, as if each of them would be updated with the same set of instances, they would all converge to similar models (Minku et al. 2010). Dynamic classifier selection is a specific case of online approach, as a pool of online learners is being maintained but only the most competent ones are being selected for the decision making process (Almeida et al. 2016).

Apart from these main trends, there exist a plethora of hybrid solutions that merge the mentioned techniques. Often dynamic ensemble line-ups are combined with online learners to achieve faster response rates (Brzeziński and Stefanowski 2014a), or online ensembles incorporate a pruning mechanism to discard classifiers that would be too difficult to properly adapt to the current state of the stream (Bifet et al. 2010b).

2.3 Imbalanced data streams

Imbalanced distribution of instances among data classes poses a significant problem for learning systems (Krawczyk 2016). This issue becomes even more challenging when being present in a data stream mining scenario (Chen and He 2013; Wang et al. 2018). Here, we must accommodate not only for skewed classes but also for the evolving nature of data. Main issues related to imbalanced data streams include (Chen and He 2013; Fernández et al. 2018):

- *Simultaneous concept and imbalance ratio drift* The proportions of objects among classes may change along the presence of concept drift. Therefore, classes are not permanently associated with their minority or majority roles, as these may change over time.
- *Evolving data characteristics* Minority instances may have a different level of hardness associated with them. This information may be used to improve the learning process by concentrating on the most difficult instances. However, in data streams these properties may change dynamically, forcing an adaptation of imbalance handling techniques.
- *Emergence and disappearance of new classes* Over time, new classes may emerge and old ones disappear. As this is usually a gradual process, it will affect the class imbalance ratios, which must be accounted for.

A real-world example of such a problem is a network of sensors that collectively work towards recognizing activities or object position. Here, the number of observations recorded by each sensor will change over time, as well as the environmental conditions in the network area. Novel activities may appear, increasing the number of classes to be recognized, as well as further changing the minority-majority relationships among classes.

Ensemble algorithms have been applied to learning from imbalanced data streams with great success. They usually aim at balancing data in every arriving chunk (Wang and Pineau 2016), or in case of online learning employing incremental sampling solutions to balance the stream instance by instance (Wang et al. 2015).

These solutions have been applied to problems that are known beforehand as imbalanced ones. However, one must note that in the data stream domain one usually does not know beforehand what characteristics of data are to be expected. While these specific solutions are effective for imbalanced streams, they are easily outperformed by other models on balanced streams (Krawczyk 2017). Class imbalance may appear periodically, e.g., after a concept drift when instances from a new concept still appear less frequently (Sun et al. 2016). Therefore, in many real-time scenarios one cannot predict if and when the stream will output imbalanced distributions. This requires classification algorithms that are able to handle effectively balanced data streams, while at the same time displaying increased robustness to class imbalance.

3 Kappa Updated Ensemble

This section presents the Kappa Updated Ensemble (KUE) algorithm, the learning model and its components, its computational and memory complexity, and its advantages as compared with state-of-the-art ensembles for data streams. KUE is detailed in Algorithm 1. The main idea of KUE is to integrate the advantages already demonstrated in the data stream mining literature of incremental learning, varying-size random subspaces, online bagging (Bifet et al. 2010b), and dynamic weighted voting (Kolter and Maloof 2007), into a single algorithm driven by the Kappa statistic while keeping a simple, effective, and computationally efficient algorithm capable of quickly self-adapting to drifts in features and data classes distribution without requiring an explicit drift detector. KUE maintains a weighted pool of diverse component classifiers and predicts the class of incoming examples by aggregating the predictions of components using weighted voting with possible abstention.

3.1 Ensemble structure and initialization

Let $\mathcal{E}$ be an ensemble classifier comprised by k components of γ base classifiers such that $\gamma_j \in \mathcal{E}$ $(j = 1, 2, \ldots, k)$. The components of the ensemble are initialized when the first data chunk $\mathcal{S}_1$ in the data stream $\mathcal{S}$ arrives. In order to promote the diversity of the ensemble components exploring feature subspaces of varied dimensionality, each base classifier γ_j is built on a different r-dimensional random subspace φ_j, where $1 \leq r \leq f$ from the original f-dimensional space in S. Importantly, the dimensionality and the subspace of features for each component are both randomized. This is a significant difference as compared to Adaptive Random Forest (Gomes et al. 2017b) which selects a fixed subspace dimensionality for all the components. We consider that allowing a different subspace dimensionality per ensemble component provides better flexibility to identify and explore more diverse random feature subspaces.

Algorithm 1 Kappa Updated Ensemble (KUE) algorithm.

```
Input: $\mathcal{S}$: data stream, $f$: number of features, $k$: number of ensemble components, $q$: number of new components to train
Output: $\mathcal{E}$: ensemble of $k$ $\gamma$ classifiers,
        $\varphi$: subspace of features for each of the $k$ components,
        $\kappa$: Kappa for each of the $k$ components
1: for $\mathcal{S}_i \in \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$ do
2:     if $\mathcal{S}_1$ then                                  ▷ Ensemble initialization
3:         for $j \in \{1, \ldots, k\}$ do
4:             $r \leftarrow$ random integer with uniform probability $[1, f]$
5:             $\varphi_j \leftarrow r$-dimensional random subspace in $\mathcal{S}_1$ where instances are weighted according to $Poisson(1)$
6:             $\gamma_j \leftarrow$ new classifier on $\varphi_j(\mathcal{S}_1)$
7:             $\kappa_j \leftarrow$ compute Kappa of $\gamma_j$ on $\varphi_j(\mathcal{S}_1)$
8:         end for
9:     else                                                     ▷ Ensemble model update
10:        for $j \in \{1, \ldots, k\}$ do
11:            $\varphi_j \leftarrow$ instances in $\mathcal{S}_i$ are weighted according to $Poisson(1)$ keeping the $r$-dimensional subspace
12:            $\gamma_j \leftarrow$ incremental train of $\gamma_j$ on $\varphi_j(\mathcal{S}_i)$
13:            $\kappa_j \leftarrow$ compute Kappa of $\gamma_j$ on $\varphi_j(\mathcal{S}_i)$
14:        end for                                              ▷ Train new components
15:        for $\{1, \ldots, q\}$ do
16:            $r \leftarrow$ random integer with uniform probability $[1, f]$
17:            $\varphi' \leftarrow r$-dimensional random subspace in $\mathcal{S}_i$ where instances are weighted according to $Poisson(1)$
18:            $\gamma' \leftarrow$ new classifier on $\varphi'(\mathcal{S}_i)$
19:            $\kappa' \leftarrow$ compute Kappa of $\gamma'$ on $\varphi'(\mathcal{S}_i)$
20:            if $\kappa' > \kappa_{min(\kappa)}$ then                ▷ Replace weakest $\gamma \in \mathcal{E}$
21:                $\varphi_{min(\kappa)} \leftarrow \varphi$'
22:                $\gamma_{min(\kappa)} \leftarrow \gamma$'
23:                $\kappa_{min(\kappa)} \leftarrow \kappa$'
24:            end if
25:        end for
26:    end if
27: end for
```

On the other hand, online bagging is applied to weight and resample with replacement instances within the subspace using the Poisson(1) distribution. It has been shown that this online bagging approach improves the performance of data stream classifiers, particularly OzaBag (Oza 2005), Leverage Bagging (Bifet et al. 2010b), and Adaptive Random Forest (Gomes et al. 2017b) follow this approach.

This way, the algorithm randomizes and diversifies the input (both instances and features) for the internal construction of the ensemble components. Once the base classifiers are trained on such data, their Kappa performances are used as weights on the voting for the class prediction of new instances.

Kappa statistic has been commonly used in imbalanced classification (Ferri et al. 2009; Jeni et al. 2013; Brzeziński et al. 2018). It evaluates the competence of a classifier by measuring the inter-rater agreement between the successful predictions and the statistical distribution of the data classes, correcting agreements that occur by mere statistical chance (Cano et al. 2013). Kappa statistic ranges from −100 (total disagreement) through 0 (default probabilistic classification) to 100 (total agreement), and it is computed as in Eq. 7:

$$Kappa = \frac{n\sum_{i=1}^{c} x_{ii} - \sum_{i=1}^{c} x_{i.}x_{.i}}{n^2 - \sum_{i=1}^{c} x_{i.}x_{.i}} \cdot 100 \tag{7}$$

where x_{ii} is the count of cases in the main diagonal of the confusion matrix (successful predictions), n is the number of examples, c is the number of classes, and $x_{.i}$, $x_{i.}$ are the column and row total counts, respectively. Importantly, Kappa penalizes all-positive or all-negative predictions, which is especially useful in multi-class imbalanced data. Moreover, since the data classes distributions may change through the progress of the stream, Kappa provides better insight than other metrics to detect changes in the performance of the algorithms due to drifts in the data classes distribution.

3.2 Ensemble model update

Every time a new data chunk $\mathcal{S}_i \in \mathcal{S}$ arrives, data is projected on the existing random subspaces for each of the ensemble components and instances are weighted through online bagging using the Poisson(1) distribution (Bifet et al. 2010b). This way, the existing components of the ensemble are incrementally updated using the new data input diversified for each member, similar to Adaptive Random Forest (Gomes et al. 2017b). The competence of the updated components is evaluated on the most current data and their Kappa are updated. This is similar to the Accuracy Updated Ensemble (Brzeziński and Stefanowski 2011) but using the Kappa statistic rather than accuracy to drive the competence of classifiers.

Two scenarios may occur here when updating the competence of the components. In case of receiving a chunk maintaining a similar data distribution than previous chunks, the performance of each of the components is expected to be stable. The components will update and refine their learned model. However, in case of a drift in the concepts, features, data classes, or noise, the performance of the components may significantly decrease, especially in the event of a sudden drift. Therefore, in order to preemptively anticipate any possible drifts of unknown nature, a new set of q classifiers are trained and evaluated each in a new r-dimensional random subspace on the most recent chunk. The variable random nature of the feature projections in the building of the new components helps to overcome drifts and noise on an undetermined sets of features. If the Kappa statistic of each of the new classifiers improves the Kappa statistic of the weakest existing component, then it replaces such component as it demonstrates to be most up to date. By replacing the weakest components with the newest classifiers, the algorithm balances the learning of new classification models and the forgetting of old classifiers which are no longer valid due to the drift in the data. This way, there is no need for an explicit drift detector as the self-update mechanism is intrinsic to the design of the ensemble update model.

3.3 Weighted voting

The class prediction $\hat{y}$ of an instance x is conducted through weighted majority voting of each of the ensemble components using their Kappa on the most current chunk. The weighted aggregated voting is defined in Eq. 8:

$$\hat{y} = \arg\max_{i} \sum_{j=1}^{k} \begin{cases} \kappa_j \; p(i \mid \gamma_j(x)) & \text{if } \kappa_j \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

This simplifies the ensemble weighting mechanism while reflecting the most up to date competence of the components. Importantly, components participate in the class voting only when their Kappa ≥ 0, i.e., those components whose competence is clearly not good enough abstain from the vote. Abstaining classifiers have demonstrated to improve the performance of online ensembles for drifting and noisy data streams (Błaszczyński et al. 2009; Krawczyk and Cano 2018). On the other hand, in the unlikely case of all classifiers having a Kappa value < 0 means that no classifier was able to model the data better than a default-hypothesis classifier based on the data class distribution. Therefore, in such cases the class prediction is returned according to a roulette selector given the class distribution frequencies.

3.4 Complexity analysis

Let us now analyze the time and memory complexity of the KUE algorithm. The algorithm receives a data chunk $\mathcal{S}_i \in \mathcal{S}$ of $|\mathcal{S}_i|$ instances. The ensemble comprises k base classifiers. The base classifier for KUE is HoeffdingTree, also known as VFDT, which builds a decision tree with a constant time and constant memory per instance (Hulten et al. 2001). Therefore, the ensemble initialization on the first chunk $\mathcal{S}_1$ has a time complexity of $\mathcal{O}(k|\mathcal{S}_1|)$. The ensemble model update on a subsequent chunk $\mathcal{S}_i$ has a time complexity of $\mathcal{O}(k|\mathcal{S}_i|)$ to update the k existing components. Moreover, the algorithm trains $q \leq k$ new components on the chunk $\mathcal{S}_i$ potentially replacing the weakest members, which has a time complexity of $\mathcal{O}(q|\mathcal{S}_i|)$. Consequently, the time complexity of KUE is $\mathcal{O}((k+q)|\mathcal{S}_i|)$.

The memory complexity of the base classifier HoeffdingTree is $\mathcal{O}(fvlc)$ where f is the number of features, v is the maximum number of values per feature, l is the number of leaves in the tree, and c is the number of classes (Hulten et al. 2001). However, KUE performs r-dimensional random subspace projections for each of the k and q components, where $r \leq f$, then effectively reducing the memory complexity of HoeffdingTree to $\mathcal{O}(rvlc)$. Therefore, the memory complexity of KUE comprising k components plus one new trained at a time is $\mathcal{O}((k+1)rvlc)$.

3.5 Contribution, novelty, and advantages over existing ensembles

Initialization of components KUE initializes the k components using data from the first chunk $\mathcal{S}_1$ projected on different random feature subspaces. On the contrary, Accuracy Updated Ensemble and Accuracy Weighted Ensemble initialize only one component in each of the initial k chunks and on whole of the feature set. This makes KUE more accurate and reliable at the beginning of the stream sequence since the k diverse components exist from the very first chunk.

Impact of learning in subspaces Online bagging and random subspaces have demonstrated to improve the performance of ensembles for online data streams, inspired by Random Forests alike methods such as Adaptive Random Forest (Gomes et al. 2017b). However, the traditional approach is to build the ensemble components on random subspaces of the same fixed dimensionality. This raises important concerns on whether this approach is the best for data streams subject to concept drifts. First, the optimal subspace size cannot be predetermined apriori as it depends on the dataset distribution and on the relevance, redundancy, or noise in the features. Second, the dimensionality of the subspace should not be constant since features and noise are subject to drift with time, making it necessary to dynamically adapt as the stream evolves. One may think about a scenario in which noise is propagating from none to all features as the stream progresses, making fixed size subspaces incapable of dynamically adapting. Therefore, the dynamic and variable size of the random subspaces in KUE constitutes a significant advantage to adapt to such scenarios. Moreover, exploring random small subspaces allows for faster model training and better classifier generalization while keeping competitive accuracy.

Kappa metric for classifier weighting Use of Kappa rather than accuracy for evaluating the competence of a data stream classifier in an ensemble is beneficial in three ways. First, there is a clear threshold for Kappa > 0 in which one can determine whether a classifier is positively contributing to the ensemble by making a likely a correct prediction. Components whose Kappa < 0 are discarded as they actually introduce misleading predictions. However, when using accuracy this is not possible since the mere accuracy value is not informative enough as it does not take into account the data classes distribution. Second, Kappa is a strict measure that will quickly drop in case of incorrect predictions, making it much more useful for weighting components rather than using accuracy that would only introduce small changes in weights. Third, the data classes distribution may drift as the stream progresses. Kappa is capable of capturing the competence of the components reflecting the possibly varying data classes distribution with time. Therefore, this is a significant advantage in KUE as compared with existing ensembles driven by accuracy such as Dynamic Weighted Majority (Kolter and Maloof 2007), Accuracy Updated Ensemble (Brzeziński and Stefanowski 2011), or Accuracy Weighted Ensemble (Wang et al. 2003).

Incorporating new classifiers Accuracy Updated Ensemble (Brzeziński and Stefanowski 2011) treats the newly trained classifier for each chunk as *perfect* and its predictions are not weighted. However, this may not be the best option, especially on complex data where high accuracy is difficult to get, making the new classifier overconfident. On the other hand, in KUE the weights of the new classifier are taken into account as soon as the classifier joins the ensemble, reflecting its most current competence.

Moreover, KUE is designed to train and replace q classifiers at a time, replacing many base classifiers if needed due to extreme drifts of the stream where all previous base classifiers would become immediately outdated. However, according to the experiments in Sect. 4.5, it is shown that training one new base classifier per chunk is sufficient to achieve competitive results while keeping the lowest computational complexity.

Abstaining classifiers Abstaining classifiers in the weighted voting is a significant advantage as compared to similar existing ensemble methods (Krawczyk and Cano 2018), especially in the case of a sudden drift where many of the components are suddenly no longer competent. In such a scenario, block-based ensembles may not react sufficiently fast to changes and it takes few iterations to learn new correct models in the new data distribution. Traditional block-based ensembles may react too slowly as classifiers generated from outdated blocks still remain valid components even though they have inaccurate weights. On the contrary, by allowing a classifier to abstain in KUE, only the components that correctly reflect the current concepts of the stream will participate in the vote, avoiding misleading predictions from outdated components.

In this work, we propose to combine them all along with the Kappa metric to lead the selection and update of the base classifiers.

4 Experimental study

This section presents the algorithms, datasets, and experiments designed to evaluate and compare the performance of the proposed method with state-of-the-art ensembles for data streams.

4.1 Algorithms

The KUE algorithm has been implemented in the Massive Online Analysis (MOA) software (Bifet et al. 2010a, 2018), which includes a collection of generators, algorithms, and performance metrics for data streams. The proposal is compared with 15 other ensemble classifiers for data stream mining available in MOA, all using a single-thread implementation in Java. These comprise a diverse set of methods including block-based, bagging, boosting, random forest, and weighted ensembles (Gama et al. 2013).

Table 1 lists the algorithms, their base learners, and their main parameters, which were selected according to the recommended values reported by their authors and other studies in this area. The KUE algorithm's source code, an executable version, and the datasets along with detailed results for the experimental analysis are available online to facilitate the reproducibility of results and future comparisons.[1] Experiments were run on an Intel Xeon CPU E5-2690v4 with 384 GB of memory on a Centos $7 \times 86 - 64$ system.

4.2 Datasets

The experimental study comprises 60 standard and 33 imbalanced data stream benchmarks to evaluate the performance of algorithms. Properties of the data stream benchmarks are presented in Tables 2 and 3. We have selected a diverse set of benchmarks reflecting various possible drifts (no drift, gradual drift, recurring drift, sudden drift), including 13 real datasets, and a variety of stream generators (RBF, RandomTree, Agrawal, AssetNegotiation, LED, Hyperplance, Mixed, SEA, Sine, STAGGER, Waveform) with different properties concerning speed and number of concept drifts. Moreover, for imbalanced datasets, we analyze the impact of the imbalance ratio (IR), including scenarios where the imbalance ratio dynamically changes through the progress of the stream. The imbalance ratio represents the relation between the number of instances of the majority class and the minority class. In the case of multi-class imbalanced dataset it is reported as the relation between the most frequent class and the least frequent class. For the sake of conducting a fair comparison, the chunk size is set to 1000 instances for all algorithms, which is the default setup in the chunk evaluation of data stream mining algorithms and it also the default value provided by MOA (Bifet et al. 2010a, 2018). To the best of our knowledge, this is one of the biggest experimental setups conducted so far as most papers in data streams are based on 8-16 benchmarks (Krawczyk et al. 2017; Gomes et al. 2017a).

4.3 Experiment 1: Evaluation on standard data streams

Table 4 shows the average performance and ranks of the classifiers on the 60 standard data streams. Results are provided for accuracy, Kappa, model update (train time), prediction (test

[1] KUE source code, datasets, detailed results, and visualizations available at: http://doi.org/10.17605/OSF.IO/6H438.

Table 1 Ensemble algorithms and their main parameters

Acronym	Algorithm's name	Parameters	Acronym	Algorithm's name	Parameters
LNSE (Elwell and Polikar 2011)	Learn^{++}. NSE	Learner: NaiveBayes ensembleSize: 15	LB (Bifet et al. 2010b)	Leveraging bagging with ADWIN	Learner: HoeffdingTree ensembleSize: 10
DWM (Kolter and Maloof 2007)	Dynamic weighted majority	Learner: NaiveBayes maxEnsembleSize: inf	SAE2 (Gomes and Enembreck 2014)	Social adaptive ensemble 2	Learner: HoeffdingTree ensembleSize: 10
DACC (Jaber et al. 2013)	Dynamic adaptation to concept changes	Learner: NaiveBayes ensembleSize: 20	AWE (Wang et al. 2003)	Accuracy weighted ensemble	Learner: HoeffdingTree ensembleSize: 10 ensembleBuffer: 30
ADACC (Jaber et al. 2013)	Anticipative and dynamic adaptation to concept changes	Learner: NaiveBayes ensembleSize: 20	AUE1 (Brzeziński and Stefanowski 2011)	Accuracy updated ensemble 1	Learner: HoeffdingTree ensembleSize: 15 ensembleBuffer: 30
OCB (Pelossof et al. 2009)	Online coordinate boosting	Learner: HoeffdingTree ensembleSize: 10	AUE2 (Brzeziński and Stefanowski 2014b)	Accuracy updated ensemble 2	Learner: HoeffdingTree ensembleSize: 10
OBA (Oza 2005)	Oza boost adwin	Learner: HoeffdingTree ensembleSize: 10	ARF (Gomes et al. 2017b)	Adaptive random forest	Learner: adaptive random forest HoeffdingTree ensembleSize: 10
OBASHT (Bifet et al. 2009)	Oza bag adaptive-size Hoeffding Tree	Learner: ASHoeffdingTree ensembleSize: 10	HEB (Van Rijn et al. 2018)	BLAST heterogeneous ensembles	Learner: HoeffdingTree ensembleSize: 5
OBAD (Bifet and Gavaldà 2007)	Oza bag adwin	Learner: HoeffdingTree ensembleSize: 10	KUE	Kappa updated ensemble	Learner: HoeffdingTree ensembleSize: 10 newcomponents: 1

Table 2 Properties of standard datasets

Dataset	Instances	Atts	Classes	IR	Properties	Dataset	Instances	Atts	Classes	IR	Properties
Powersupply	29,928	3	24	1	Unknown drift	AssetNegotiation-F5	1,000,000	5	2	1	No drift
Electricity	45,312	8	2	1	Unknown drift	Hyperplane-drift	1,000,000	10	4	1	Gradual drift
Shuttle	58,000	9	7	13	Unknown drift	LED	1,000,000	7	10	1	No drift
Connect-4	67,557	43	3	6	Unknown drift	LED-noise	1,000,000	7	10	1	Noise
Census	299,284	42	2	15	Unknown drift	LED-drift	1,000,000	7	10	1	Gradual drift
CovType	581,012	54	7	29	Unknown drift	LED-drift-4	1,000,000	7	10	1	Sudden drift
Poker	829,201	11	10	10	Unknown drift	Mixed-imbalanced	1,000,000	5	2	1	Sudden drift
BNG_bridges	1,000,000	13	6	4	Unknown drift	Mixed-balanced	1,000,000	5	2	1	Sudden drift
BNG_lymph	1,000,000	19	4	17	Unknown drift	RBF	1,000,000	50	4	1	No drift
BNG_zoo	1,000,000	17	7	9	Unknown drift	RBF-drift	1,000,000	50	4	1	Gradual drift slow
BNG_wine	1,000,000	14	3	1	Unknown drift	RBF-drift-fast	1,000,000	50	4	1	Gradual drift fast
BNG_hepatitis	1,000,000	20	2	4	Unknown drift	RBF-drift-gradual	1,000,000	50	4	1	Gradual drift
IntelLabSensors	2,313,153	6	58	1	Unknown drift	RBF-drift-recu	1,000,000	50	4	1	Recurrent drift
Agrw-F1	1,000,000	9	2	1	No drift	RandomTree	1,000,000	10	2	1	No drift
Agrw-F10toF1-drift	1,000,000	9	2	1	Gradual drift	RandomTree-drift	1,000,000	10	2	1	Gradual drift
Agrw-F1toF10-drift	1,000,000	9	2	1	Gradual drift	RandomTree-drift-recu	1,000,000	10	2	1	Recurrent drift
Agrw-F1toF10-drift-slow	1,000,000	9	2	1	Gradual drift slow	RandomTree-drift-fast	1,000,000	10	2	1	Gradual drift fast
Agrw-F1toF10-drift-fast	1,000,000	9	2	1	Gradual drift fast	SEA-F1	1,000,000	3	2	1	No drift
Agrw-F1toF10-drift-sudden	1,000,000	9	2	1	Sudden drift	SEA-F2	1,000,000	3	2	1	No drift
Agrw-F2F4F6F8F1F3-drift	1,000,000	9	2	1	Sudden drift	SEA-F4	1,000,000	3	2	1	No drift
Agrw-F3F5F3F5F3F5-drift	1,000,000	9	2	1	Recurrent drift	SEA-drift-suddent	1,000,000	3	2	1	Sudden drift
Agrw-F3F5F7F3F5F7-drift	1,000,000	9	2	1	Recurrent drift	SEA-drift-fast	1,000,000	3	2	1	Gradual drift fast
Agrw-F3F5F7F5F3-drift	1,000,000	9	2	1	Sudden drift	Sine-F1	1,000,000	5	2	1	No drift

Table 2 continued

Dataset	Instances	Atts	Classes	IR	Properties	Dataset	Instances	Atts	Classes	IR	Properties
Agrw-F7toF2-drift	1,000,000	9	2	1	Gradual drift	Sine-F2	1,000,000	5	2	1	No drift
Agrw-F9F7F3F5F4F2-drift	1,000,000	9	2	1	Sudden drift	Sine-F3	1,000,000	5	2	1	No drift
Agrw-F-random-drift	1,000,000	9	2	1	Sudden drift	STAGGER-F1	1,000,000	3	2	1	No drift
AssetNegotiation-F1	1,000,000	5	2	1	No drift	STAGGER-F2	1,000,000	3	2	1	No drift
AssetNegotiation-F2	1,000,000	5	2	1	No drift	STAGGER-F1toF3-drift	1,000,000	3	2	1	Gradual drift
AssetNegotiation-F3	1,000,000	5	2	1	No drift	Waveform	1,000,000	40	3	1	No drift
AssetNegotiation-F4	1,000,000	5	2	1	No drift	Waveform-drift	1,000,000	40	3	1	Gradual drift

Table 3 Properties of imbalanced datasets

Dataset	Instances	Atts	Classes	IR	Dataset	Instances	Atts	Classes	IR
Agrw-IR-5	1,000,000	9	2	5	SEA-IR-5	1,000,000	3	2	5
Agrw-IR-10	1,000,000	9	2	10	SEA-IR-10	1,000,000	3	2	10
Agrw-IR-20	1,000,000	9	2	20	SEA-IR-20	1,000,000	3	2	20
Agrw-IR-100	1,000,000	9	2	100	SEA-IR-100	1,000,000	3	2	100
AssetNegotiation-IR-5	1,000,000	5	2	5	Agrw-F1toF10-IR-1to20	1,000,000	9	2	1 to 20
AssetNegotiation-IR-10	1,000,000	5	2	10	AssetNeg-F1toF5-IR-1to10	1,000,000	5	2	1 to 10
AssetNegotiation-IR-20	1,000,000	5	2	20	RBF-drift-IR-1to10	1,000,000	50	2	1 to 10
AssetNegotiation-IR-100	1,000,000	5	2	100	SEA-F1toF4-IR-1to5	1,000,000	3	2	1 to 5
Hyperplane-IR-5	1,000,000	10	2	5	Agrw-F1toF10-IR-10to1	1,000,000	9	2	10 to 1
Hyperplane-IR-10	1,000,000	10	2	10	AssetNeg-F1toF5-IR-10to10	1,000,000	5	2	10 to 1 to 10
Hyperplane-IR-20	1,000,000	10	2	20	Poker-1-2vsAll	1,000,000	11	2	13
Hyperplane-IR-100	1,000,000	10	2	100	IntelLabSensors-1to9vsAll	2,313,153	6	2	7
RBF-IR-5	1,000,000	10	2	5	IntelLabSensors-1to5vsAll	2,313,153	6	2	13
RBF-IR-10	1,000,000	10	2	10	IntelLabSensors-1to3vsAll	2,313,153	6	2	17
RBF-IR-20	1,000,000	10	2	20	IntelLabSensors-1vsAll	2,313,153	6	2	54
RBF-IR-100	1,000,000	10	2	100	CovType-1-2vsAll	581,012	54	2	7
					BNG_bridges-1vsAll	1,000,000	13	2	7

Table 4 Performance of ensemble classifiers on standard data streams: average and rank on 60 streams

Metric	LNSE	DWM	DACC	ADACC	OCB	OBA	OBASHT	OBAD	LB	SAE2	AWE	AUE1	AUE2	ARF	HEB	KUE
Avg. accuracy	71.00	75.51	69.75	69.84	68.41	79.14	82.93	84.55	84.49	79.10	76.97	83.50	83.67	83.93	82.80	**85.90**
Avg. Kappa	50.69	58.29	48.93	49.14	47.91	65.16	70.77	73.62	73.60	64.45	61.87	71.56	73.23	72.58	70.36	**76.19**
Avg. train time (s)	3.5988	**0.0123**	0.0197	0.0522	0.0555	0.7407	0.0453	0.2668	1.5418	0.0320	0.1686	0.3709	0.0838	0.1731	0.2844	0.0597
Avg. test time (s)	0.2271	0.0104	**0.0046**	0.0076	0.0156	0.0159	0.0213	0.0136	0.0170	0.0118	0.0178	0.0162	0.0172	0.0167	0.0492	0.0157
Avg. RAM-Hours	3.2E−1	**1.1E−7**	6.5E−7	6.5E−6	6.8E−4	5.8E−2	2.4E−4	4.3E−3	6.6E−2	9.2E−5	2.1E−4	3.0E−3	4.7E−4	4.9E−3	2.8E−4	8.2E−4
Rank accuracy	12.89	11.40	14.67	14.12	11.29	9.79	6.94	3.76	4.71	11.05	10.07	6.62	4.08	5.01	6.75	**2.86**
Rank Kappa	12.80	11.16	14.72	14.11	11.38	9.77	7.07	3.75	4.66	10.87	10.02	6.82	4.14	5.14	6.72	**2.88**
Rank train time	15.47	**1.15**	2.17	5.16	6.52	11.53	4.85	10.72	14.58	3.45	10.02	12.47	7.68	11.18	12.43	6.63
Rank test time	15.09	4.85	**2.56**	4.80	7.80	9.36	12.03	7.51	10.12	5.88	9.19	8.10	9.34	11.18	8.43	7.77
Rank RAM-Hours	15.32	**1.00**	2.02	3.35	7.93	13.48	6.37	10.88	13.98	5.33	5.73	10.87	7.78	12.85	9.72	7.38
Meta-rank	14.31	5.91	7.23	8.31	8.98	10.79	7.45	7.32	9.61	7.32	9.01	8.97	6.60	9.07	8.81	**5.50**

Bold values represent the best results

time), and memory consumption (RAM-Hours). Different metrics provide complementary perspectives of the predictive power of the classifiers. Ranks of the classifiers according to Friedman are reported for each metric. Finally, the meta-rank shows the average rank across all metrics.

Table 5 shows the accuracy for each of the 60 standard data streams. The accuracy is averaged through all the instances of the stream. Detailed results for all metrics and datasets are available online (see footnote 1) including plots with the variation of the performance metrics through the progress of the stream.

Table 6 presents the outcomes of Wilcoxon statistical test (García and Herrera 2008; García et al. 2010) where the lower the p value the bigger differences between the algorithms, while Figs. 1 and 2 depict the visualizations of ranks according to the Bonferroni–Dunn test. Figure 3 presents the pairwise comparison between KUE and reference methods with the respect to the number of wins, ties, and loses on all datasets. Figure 4 depicts the distribution of the frequencies of ranks achieved by all of the ensemble classifiers over all datasets.

Finally, Figs. 5, 6 and 7 present detailed results over all processed instances in the stream for three selected datasets with respect to accuracy, Kappa, chunk update time, and memory consumption.

Comparison with other ensembles KUE has been compared with 15 other ensemble classifiers on 60 standard data stream benchmarks. KUE is capable of outperforming in a statistically significant manner 11 out of 15 reference classifiers on more than 45 datasets each. Therefore, we will focus on a detailed analysis of the top 4 reference methods, which are OBAD, LB, AUE2, and ARF.

OBAD returns the best performance on average among all classifiers for both accuracy and Kappa metrics. From Fig. 3 one can see that OBAD returns better results than KUE on 15 datasets, and ties with KUE on 20 datasets. Figure 4 shows that OBAD does not score the first rank as frequently as LB or KUE but at the same time is rarely in a lower position than 5th. This makes it the most challenging reference method for KUE, as OBAD proves to be a good all-purpose classifier. However, the superiority of KUE can be seen on Fig. 4, as KUE never achieves a lower rank than 6th, while for certain datasets OBAD can position itself on as low ranks as 9th or 10th. Additionally, the pairwise statistical analysis proves that differences between KUE and OBAD are statistically significant.

LB is another challenging reference classifier. It is interesting to see that according to Fig. 3, LB wins with KUE on a higher number of datasets than OBAD. At the same time, KUE wins with LB more frequently than with OBAD, as ties here are very rare. This shows that LB delivers unstable results that are highly dependent on the datasets. Analyzing Fig. 4, we can see that LB scores a similar number of first positions as KUE, and more than OBAD. At the same time, there are certain datasets on which LB achieves much lower scores, ultimately leading to its lower overall average ranking. This shows that LB is not as data-invariant as KUE, making it less reliable for general data stream mining purposes.

AUE2 is a natural counterpart of KUE, as they share similar roots in how to expand the ensemble and use a predictive metric for weighting. Results clearly point to the superiority of KUE over AUE2, which can be contributed to our proposed mechanisms. By using random feature subsets we achieved a better multi-view grasp of data stream characteristics, and by adding new classifiers only when they contribute to the ensemble we are able to avoid updating the ensemble set-up when not necessary. Figure 3 shows that KUE outperforms AUE2 on a similar number of datasets as it outperforms OBAD but is much more likely to tie with AUE2 than lose to it. When analyzing the rank frequencies, one can see that AUE2 often scores as the second best classifier, incapable of securing the first rank on most of the datasets.

Table 5 Accuracy of ensemble classifiers on each of the standard data streams

Accuracy	LNSE	DWM	DACC	ADACC	OCB	OBA	OBASHT	OBAD	LB	SAE2	AWE	AUE1	AUE2	ARF	HEB	KUE
Powersupply	4.17	13.03	7.23	7.23	2.09	4.17	15.61	15.60	**16.56**	14.25	15.40	15.40	15.32	13.54	14.71	15.99
Electricity	59.31	70.98	56.34	58.30	74.50	74.70	76.28	76.06	75.99	72.08	70.72	73.13	76.55	76.86	69.84	**76.90**
Shuttle	93.79	89.91	92.03	92.11	74.21	99.67	97.78	99.13	**99.80**	90.24	97.66	98.35	98.96	99.74	98.79	99.41
Connect-4	58.32	67.01	61.05	61.35	69.12	67.59	70.53	67.72	70.21	62.08	62.06	66.04	70.08	67.49	68.96	**72.17**
Census	84.14	91.40	90.37	92.23	93.47	93.10	**94.66**	94.34	94.04	90.13	92.98	93.81	94.33	94.12	92.33	**94.66**
CovType	69.97	71.96	61.70	55.89	71.29	85.73	80.79	84.26	**90.11**	76.21	80.03	83.01	86.68	86.24	88.27	87.53
Poker	54.55	57.81	54.53	54.69	74.62	84.35	72.19	63.17	**90.99**	56.78	60.01	60.67	72.06	68.98	71.01	90.37
BNG_bridges	68.59	70.22	59.02	59.02	20.44	57.27	69.36	51.59	43.12	44.19	46.33	56.40	62.61	68.19	**74.31**	68.18
BNG_lymph	84.51	83.79	80.21	80.42	55.19	88.53	89.73	91.44	**91.52**	84.20	87.04	90.47	91.20	91.47	90.26	91.28
BNG_zoo	90.23	91.74	87.24	87.35	58.67	91.83	92.49	92.35	87.04	84.61	69.77	91.90	91.98	**93.78**	92.72	92.53
BNG_wine	91.34	90.99	87.01	87.12	68.49	93.03	93.23	94.41	**94.75**	87.80	92.59	92.37	93.60	94.32	93.21	94.30
BNG_hepatitis	87.05	86.31	83.97	84.41	90.63	87.90	90.01	92.39	**92.77**	87.87	88.86	90.64	92.23	92.29	91.50	92.25
IntelLabSensors	1.82	87.45	95.28	93.52	3.75	54.30	95.20	97.73	97.79	87.55	3.84	**98.01**	3.63	97.86	96.66	**98.01**
Agrw-F1	82.32	87.54	81.05	83.21	93.63	90.72	94.63	94.85	94.33	90.45	94.27	94.56	94.95	93.82	94.92	**94.98**
Agrw-F10toF1-drift	76.48	76.98	75.19	75.27	64.30	77.50	82.66	86.42	84.21	82.35	83.47	87.05	88.58	85.58	77.48	**89.68**
Agrw-F1toF10-drift	68.42	70.78	66.27	66.67	88.23	79.80	88.67	90.93	89.03	87.09	83.49	89.05	**92.68**	89.03	85.79	91.79
Agrw-F1toF10-drift-slow	76.53	77.26	75.06	75.35	79.46	79.81	85.75	88.29	85.40	83.63	83.36	86.85	**90.29**	86.58	82.80	90.16
Agrw-F1toF10-drift-fast	75.24	76.89	73.58	73.89	78.10	77.33	83.19	85.90	82.56	83.16	82.06	85.35	**87.61**	83.43	80.31	87.11
Agrw-F1toF10-drift-sudden	76.64	77.71	75.26	75.55	79.48	80.76	84.91	88.66	86.22	84.86	83.89	86.89	**90.31**	87.04	82.87	89.35
Agrw-F2F4F6F8F1F3-drift	72.98	74.76	71.94	72.18	79.60	76.87	81.03	85.76	84.96	83.15	82.92	88.61	89.11	85.70	78.75	**90.45**
Agrw-F3F5F3F5F3F5-drift	61.51	64.22	58.59	58.59	74.08	78.61	79.66	83.15	81.05	79.60	79.82	80.96	84.41	84.22	76.32	**88.53**
Agrw-F3F5F7F3F5F7-drift	68.93	70.87	65.94	66.15	69.68	83.32	81.75	86.51	83.58	82.53	82.56	84.18	86.97	85.27	80.02	**89.35**
Agrw-F3F5F7F5F3-drift	66.55	68.84	63.76	63.76	65.84	82.29	80.18	85.84	83.01	84.54	81.46	83.21	86.31	86.06	79.02	**89.24**

Table 5 continued

Accuracy	LNSE	DWM	DACC	ADACC	OCB	OBA	OBASHT	OBAD	LB	SAE2	AWE	AUE1	AUE2	ARF	HEB	KUE
Agrw-F7toF2-drift	76.38	78.23	72.60	73.01	82.81	77.45	83.67	84.44	81.64	81.68	80.55	83.60	85.00	**88.39**	85.16	84.48
Agrw-F9F7F3F5F4F2-drift	71.41	72.69	69.09	69.12	82.55	78.15	82.30	85.49	83.61	82.47	81.91	86.05	87.67	85.22	77.03	**88.99**
Agrw-F-random-drift	76.50	77.46	75.38	75.46	74.74	76.28	79.70	82.88	81.60	80.36	80.67	83.84	86.15	82.95	78.42	**88.16**
AssetNegotiation-F1	85.91	84.05	82.23	82.71	93.69	90.66	93.93	**94.16**	93.99	89.72	94.11	94.04	**94.16**	94.05	94.15	**94.16**
AssetNegotiation-F2	92.70	92.11	90.70	90.73	94.69	92.43	94.84	**94.89**	94.76	90.36	94.84	94.39	94.88	94.85	94.87	**94.89**
AssetNegotiation-F3	90.94	91.56	87.94	87.92	94.60	92.05	94.47	**94.81**	94.65	90.27	94.46	94.70	94.80	94.76	94.80	**94.81**
AssetNegotiation-F4	91.48	92.06	88.50	88.49	94.48	91.96	94.41	**94.72**	94.47	90.21	94.43	94.61	94.71	94.64	94.70	94.71
AssetNegotiation-F5	92.61	92.38	90.55	90.44	94.68	92.43	94.83	**94.90**	94.84	90.41	94.75	94.65	**94.90**	94.85	94.89	**94.90**
Hyperplane-drift	86.20	88.06	80.66	81.16	85.78	76.26	88.65	88.01	86.81	82.67	86.00	86.51	88.61	84.52	**92.57**	87.40
LED	67.84	71.15	48.27	48.35	17.44	73.62	73.95	73.94	73.80	67.60	73.94	**73.96**	73.95	73.78	73.88	**73.96**
LED-noise	30.88	37.65	20.96	20.96	5.73	10.01	**41.54**	41.52	38.62	37.60	**41.54**	41.16	41.51	41.35	41.52	**41.54**
LED-drift	40.54	50.22	32.71	32.71	12.56	41.76	53.15	53.23	51.31	48.53	51.70	52.95	**53.32**	52.59	52.93	53.17
LED-drift-4	67.84	71.15	48.36	48.42	17.44	73.63	73.95	73.94	73.80	67.60	73.94	73.96	73.95	73.87	73.89	**73.97**
Mixed-imbalanced	90.76	91.46	88.61	88.73	98.94	99.67	98.15	99.53	**99.79**	93.41	93.54	99.35	99.42	99.76	98.34	99.48
Mixed-balanced	90.91	91.19	89.16	89.44	98.98	99.67	98.28	99.50	**99.79**	93.16	93.50	99.32	99.39	99.76	98.31	99.51
RBF	70.28	70.43	65.01	65.04	92.08	93.00	89.41	94.82	**95.10**	89.16	60.62	94.62	94.73	94.55	92.63	94.75
RBF-drift	51.14	52.59	50.54	50.54	55.16	51.33	54.13	**56.35**	54.89	53.88	50.16	51.39	52.80	54.61	54.17	55.61
RBF-drift-fast	27.62	29.80	26.95	26.95	25.39	31.32	33.84	29.79	28.51	32.34	27.85	30.68	31.20	33.55	32.89	**34.24**
RBF-drift-gradual	72.87	74.96	61.91	61.93	49.62	96.51	94.80	97.54	**98.16**	88.48	79.15	96.27	97.23	97.38	97.78	97.26
RBF-drift-recu	62.41	63.79	49.42	49.42	48.88	96.43	93.39	97.40	**97.80**	84.45	68.17	96.88	97.06	97.03	93.97	96.66
RandomTree	51.60	56.19	37.93	37.93	37.97	93.53	87.57	95.73	**97.74**	85.43	76.84	95.23	95.34	89.54	91.90	95.35
RandomTree-drift	46.81	57.29	38.17	38.17	51.92	62.12	83.36	91.93	91.45	80.05	60.24	69.59	**93.50**	53.94	86.25	92.41

Table 5 continued

Accuracy	LNSE	DWM	DACC	ADACC	OCB	OBA	OBASHT	OBAD	LB	SAE2	AWE	AUE1	AUE2	ARF	HEB	KUE
RandomTree-drift-recu	45.23	49.44	34.33	34.33	22.07	72.04	66.19	83.70	**86.37**	72.34	61.85	83.21	83.70	80.07	62.10	81.89
RandomTree-drift-fast	43.25	44.23	32.45	32.45	22.64	53.22	67.65	81.26	**84.77**	63.30	50.52	80.87	80.79	79.61	65.26	75.88
SEA-F1	85.84	87.21	83.62	84.02	88.03	84.60	89.30	89.77	89.68	85.60	82.30	86.52	89.13	**89.82**	89.49	89.78
SEA-F2	85.56	87.03	83.22	83.70	88.68	84.29	89.32	89.77	**89.84**	85.59	87.68	89.35	89.74	89.79	89.45	89.74
SEA-F4	85.30	87.69	83.37	83.64	88.68	85.41	89.29	89.76	**89.84**	85.60	87.96	89.33	89.71	89.79	89.35	89.77
SEA-drift-suddent	85.77	86.93	83.73	83.90	88.23	83.59	88.03	89.00	89.46	85.11	85.62	88.80	89.00	**89.58**	88.27	88.98
SEA-drift-fast	85.05	85.01	82.94	82.89	87.93	82.03	87.30	88.14	89.18	84.21	86.22	87.81	88.36	**89.35**	88.20	88.35
Sine-F1	92.55	93.30	92.20	92.53	99.51	99.72	99.47	99.83	**99.86**	94.51	97.99	99.80	99.83	99.67	99.59	99.81
Sine-F2	92.55	93.34	92.20	92.53	99.48	99.71	99.48	99.83	**99.85**	94.57	97.99	99.79	99.83	99.67	99.59	99.81
Sine-F3	83.22	82.37	82.05	81.88	99.05	99.57	95.44	99.59	**99.77**	93.58	92.25	99.50	99.53	99.24	98.54	99.55
STAGGER-F1	89.82	**100**	99.96	99.96	**100**	**100**	**100**	**100**	**100**	95.04	96.11	99.98	99.99	**100**	**100**	**100**
STAGGER-F2	44.40	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	95.02	99.96	99.96	99.98	**100**	**100**	**100**
STAGGER-F1toF3-drift	72.35	72.41	69.33	69.51	73.32	67.21	74.70	**75.11**	74.88	71.87	72.29	74.21	75.03	74.81	74.99	75.10
Waveform	80.19	78.39	73.59	73.65	55.05	78.37	83.57	**85.52**	84.97	80.18	81.13	82.94	85.33	83.60	83.75	85.29
Waveform-drift	80.19	78.42	73.70	73.72	55.16	79.46	83.45	**85.51**	85.00	80.06	81.13	83.25	85.27	83.42	83.60	85.15
Avg. accuracy	71.00	75.51	69.75	69.84	68.41	79.14	82.93	84.55	84.49	79.10	76.97	83.50	83.67	83.93	82.80	**85.90**

Bold values represent the best results

Table 6 Wilcoxon test for standard data streams

Metric KUE versus	Accuracy *p* value	Kappa *p* value	Metric KUE versus	Accuracy *p* value	Kappa *p* value
LNSE	1.8E−11	1.8E−11	LB	1.2E−02	1.2E−02
DWM	4.4E−11	1.2E−10	SAE2	1.7E−11	1.7E−11
DACC	2.5E−11	2.5E−11	AWE	2.5E−11	1.8E−11
ADACC	2.5E−11	2.5E−11	AUE1	4.9E−09	1.9E−09
OCB	6.0E−12	5.5E−11	AUE2	8.9E−03	5.2E−03
OBA	8.7E−11	1.2E−10	ARF	9.4E−05	4.4E−05
OBASHT	1.1E−09	4.8E−10	HEB	7.1E−08	5.7E−08
OBAD	1.8E−02	1.9E−02			

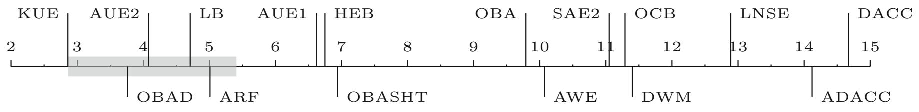

Fig. 1 Bonferroni–Dunn test for accuracy on standard data

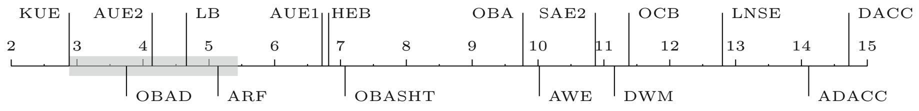

Fig. 2 Bonferroni–Dunn test for Kappa on standard data

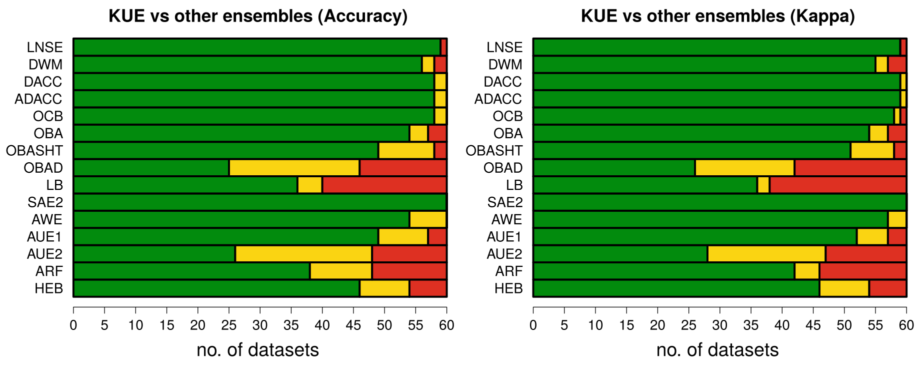

Fig. 3 Comparison of KUE and reference ensemble classifiers with respect to the number of wins (green), ties (yellow), and losses (red) over 60 standard data stream benchmarks datasets. A tie was considered, when the difference in obtained metric values were ≤ 0.05 (Color figure online)

ARF is surprisingly the weakest of all top four methods. It wins with KUE on the same number of datasets as AUE2 but is less likely to tie with it. Therefore, ARF loses to KUE most frequently of all four top performing classifiers. Analyzing its rank frequencies shows that they are evenly distributed between second and 11th rank, proving that ARF is subject to the highest variance in its performance.

These results allow us to conclude that KUE is a suitable choice for a wide array of data stream mining problems, always returning a satisfactory performance. Additionally,

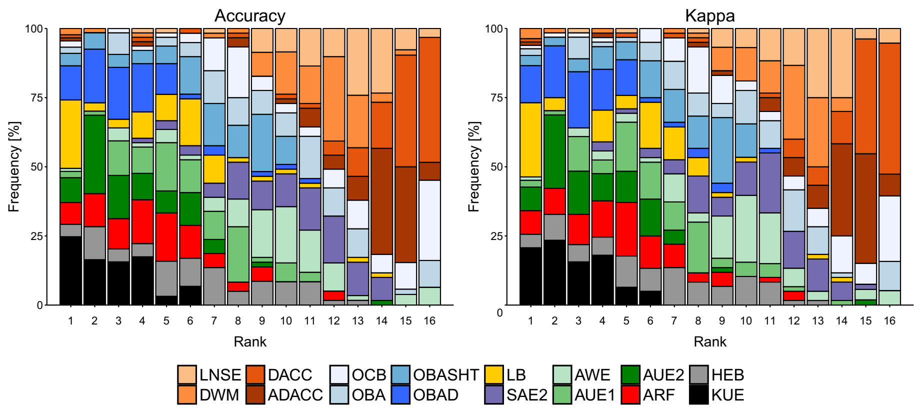

Fig. 4 Frequencies of ranks scored by ensemble classifiers on 60 standard data stream benchmarks

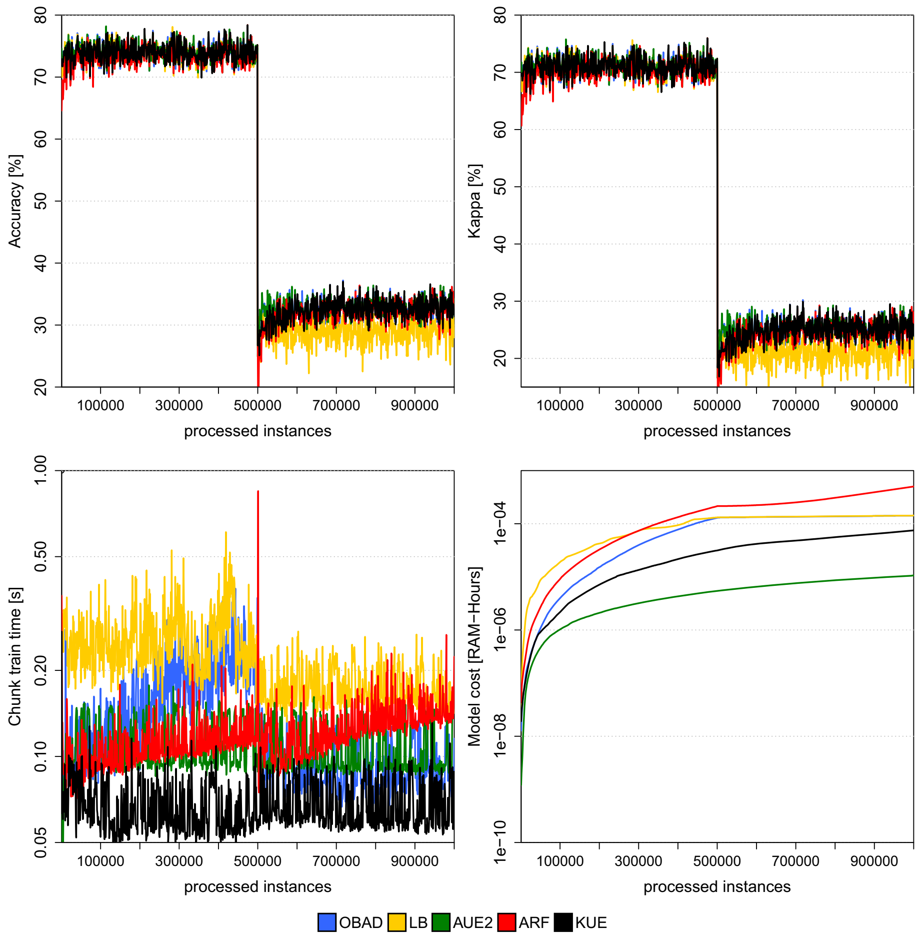

Fig. 5 Performance of top 5 ensemble methods according to their prequential accuracy, prequential Kappa, chunk update time, and memory consumption on LED generator with sudden concept drift

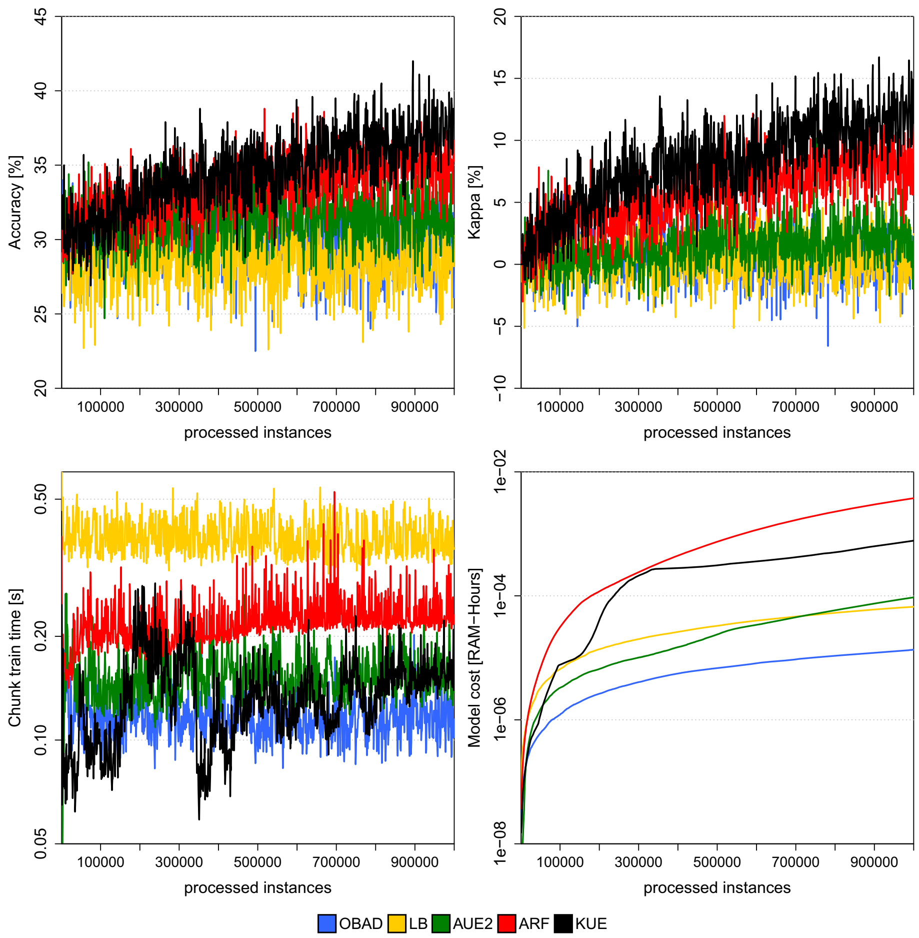

Fig. 6 Performance of top 5 ensemble methods according to their prequential accuracy, prequential Kappa, chunk update time, and memory consumption on RBF generator with gradual concept drift

the analysis of rank frequencies shows its high stability over all datasets, making KUE an excellent off-the-shelf algorithm.

Computational complexity KUE offers both better predictive power and lower time complexity. Not only does KUE outperform in both accuracy and Kappa metrics the top four reference ensemble methods (OBAD, LB, AUE2, and ARF), but it is also characterized by up to 10 times faster update time per chunk and using an order of magnitude less memory. This shows that the proposed components of KUE are not only lightweight themselves but also lead to gaining an advantage over reference ensembles.

Using feature subspaces speeds up the training of new classifiers, as the models are fitted in a lower dimensional space. Additionally, the classifier selection procedure does not impose any additional cost, as it simply measures the Kappa metric returned by new and existing classifiers.

Figures 5, 6 and 7 allow us to analyze the time and memory requirements of KUE in more details over three selected data streams. KUE displays a very stable utilization of computa-

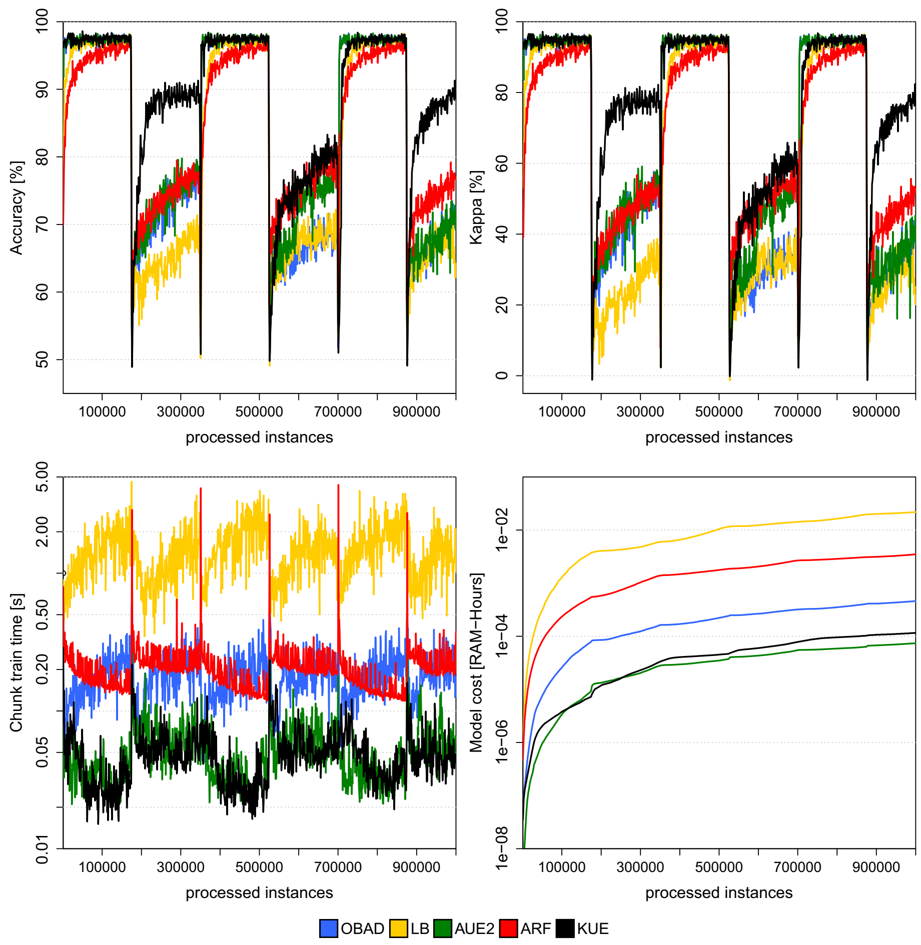

Fig. 7 Performance of top 5 ensemble methods according to their prequential accuracy, prequential Kappa, chunk update time, and memory consumption on Agrawal generator with recurrent sudden concept drift

tional resources that does not display any significant variations over time. Furthermore, KUE complexity is not influenced by the sudden presence of concept drift as can be seen for LED and Agrawal streams.

All these characteristics show that KUE combines an accurate predictive power with low computational resource consumption, making it a suitable choice for high-speed data stream mining, as well as for applications in environments with constrained resources, like edge computing on mobile devices.

Recovery rates after concept drift Recovery after a concept drift is crucial in every data stream mining algorithm. It can be seen as a period of time (or a number of instances) after which a classifier is capable of returning a stable performance, i.e., capturing the properties of a new concept. This is especially important in case of a sudden change where base classifiers need to be trained from scratch. The recovery period is a time in which classifiers cannot be treated as a competent and thus the risk of making an incorrect prediction is increased.

A popular way of analyzing the recovery rates is a visual analysis of error rates by plotting the performance over an entire stream. This can be seen in Figs. 5, 6 and 7 for three selected datasets: LED and Agrawal with a sudden concept drift, and RBF with a gradual concept drift. Each of these datasets was prepared in such a way that clearly emphasizes the point of change. LED has a single drift after 500,000 instances, Agrawal has 5 drifts after 175,000 instances each, and RBF has a drift present through the entire time. This allows us to analyze the behavior of KUE and the top 4 reference ensembles on these challenging scenarios.

One can see that in all three cases KUE is capable of reducing its error in the smallest amount of time. AUE2 and ARF are capable of satisfactory drift recovery, yet still being slower than KUE in most of the cases. OBAD and LB require the highest number of instances to reduce their error, invalidating them for high-speed data streams with frequent rapid changes.

The excellent adaptability of KUE can be contributed to two factors: usage of feature subspaces by each classifier, and weighting base classifiers according to Kappa metric. The former property offers interesting behavior on drifting streams, as only certain features may be affected by concept drift. KUE holds a pool of diverse base classifiers, each using a different subset of features. This allows them to better anticipate the direction of changes and improves the probability of having a classifier that uses features less (or not at all) affected by concept drift. The latter property offers capabilities for boosting the importance of the most competent classifiers after concept drift presence. The Kappa metric promotes classifiers that are most different from random decisions, thus allowing to assign them the highest weight in the class prediction. This naturally combines with the fact that classifiers use different features, allowing KUE to focus on classifiers that were least affected by concept drift, or that are achieving the best recovery rates using new instances.

4.4 Experiment 2: Evaluation on imbalanced data streams

The aim of the second experiment was to examine the robustness of KUE to class imbalance, as compared to the reference ensemble algorithms. While KUE was not specifically designed for the imbalanced data stream mining, we wanted to evaluate how KUE will respond to skewed class distributions, especially when combined with the concept drift. We do not compare KUE with specific methods designed for imbalanced data streams (Brzeziński and Stefanowski 2018), as our work does not focus on this issue. We wanted to check if, by simple alteration of the general streaming ensemble scheme and the Kappa statistic, we are able to improve the performance on imbalanced data without using dedicated sampling or algorithm-level modifications.

Table 7 shows the average performance and ranks of the classifiers on the 33 imbalanced data streams. Results are provided for accuracy, AUC, Kappa, G-Mean, model update (train time), prediction (test time), and memory consumption (RAM-Hours). Different metrics provide complementary perspectives of the predictive power of the classifiers. Ranks of the classifiers according to Friedman are reported for each metric. Finally, the meta-rank shows the average rank across all metrics.

Table 8 shows the Kappa for each of the 33 standard data streams. The Kappa is averaged through all the instances of the stream. Detailed results for all metrics and datasets are available online[1] including plots with the variation of the performance metrics through the progress of the stream.

Table 9 presents the outcomes of Wilcoxon test (García and Herrera 2008; García et al. 2010) where the lower the p value the bigger differences between the algorithms, while Figs. 8, 9, 10 and 11 depict the visualizations of ranks according to the Bonferroni–Dunn

Table 7 Performance of ensemble classifiers on imbalanced data streams: average and rank on 33 streams

Metric	LNSE	DWM	DACC	ADACC	OCB	OBA	OBASHT	OBAD	LB	SAE2	AWE	AUE1	AUE2	ARF	HEB	KUE
Avg. accuracy	78.74	91.08	89.31	89.24	93.46	92.38	93.40	94.53	94.29	89.74	80.73	93.86	83.25	93.98	94.26	**94.62**
Avg. AUC	71.45	69.54	75.88	77.77	80.50	84.23	87.39	87.37	85.56	76.83	72.36	84.82	80.11	86.62	81.32	**87.86**
Avg. Kapppa	30.43	41.93	38.11	36.86	64.40	58.58	56.52	67.32	64.84	49.06	31.01	59.07	51.70	62.63	67.85	**69.84**
Avg. G-Mean	46.07	52.91	56.12	55.56	75.89	71.04	65.60	75.05	72.60	71.42	37.55	66.51	59.18	71.01	76.72	**77.18**
Avg. train time (s)	4.3963	**0.0114**	0.0230	0.1298	0.0790	5.4686	0.0426	0.4967	3.3764	0.0344	0.0951	0.3907	0.0940	0.2084	0.4338	0.0554
Avg. test time (s)	0.6892	0.0099	0.0108	0.0422	0.0148	0.0241	0.0147	0.0203	0.0294	0.0097	**0.0073**	0.0127	0.0147	0.0248	0.0616	0.0157
Avg. RAM-Hours	5.2E−1	**7.0E−8**	4.3E−7	1.8E−5	8.8E−4	3.2E−1	1.1E−4	5.4E−3	1.4E−1	9.4E−5	1.4E−5	2.7E−3	3.0E−4	6.3E−3	3.3E−4	4.3E−4
Rank accuracy	13.29	11.08	14.32	13.65	8.17	9.68	7.29	3.24	4.18	13.76	10.67	6.80	5.68	5.42	5.65	**3.12**
Rank AUC	12.42	14.73	12.94	11.98	10.36	8.36	4.12	3.17	5.20	13.15	10.27	5.56	5.52	5.50	9.91	**2.80**
Rank Kappa	13.18	12.48	13.20	13.02	6.65	8.30	8.33	3.55	4.35	11.48	12.41	8.06	6.45	5.73	5.77	**3.03**
Rank G-Mean	12.64	12.52	12.68	12.89	6.24	7.68	9.32	3.83	4.74	9.18	12.70	8.89	7.05	6.15	5.89	**3.59**
Rank train time	15.09	**1.09**	2.61	8.03	7.45	13.64	4.06	11.79	14.64	3.36	7.58	11.00	7.64	10.55	12.12	5.36
Rank test time	14.21	4.83	5.44	10.38	7.45	11.03	8.06	10.71	13.11	4.68	**4.08**	7.15	8.35	12.36	6.14	8.02
Rank RAM-Hours	15.45	**1.00**	2.14	5.55	8.73	14.11	6.36	11.29	14.03	4.44	5.03	10.21	7.55	12.27	9.15	8.70
Meta-rank	13.76	8.25	9.05	10.79	7.87	10.40	6.79	6.80	8.61	8.58	8.96	8.24	6.89	8.28	7.81	**4.95**

Bold values represent the best results

Table 8 Kappa of ensemble classifiers on each of the imbalanced data streams

Kappa	LNSE	DWM	DACC	ADACC	OCB	OBA	OBASHT	OBAD	LB	SAE2	AWE	AUE1	AUE2	ARF	HEB	KUE
Agrw-IR-5	34.46	7.70	35.97	45.67	81.95	76.89	85.23	86.99	85.54	73.95	85.52	86.65	87.03	83.40	87.00	**87.20**
Agrw-IR-10	14.33	0.77	19.10	26.15	77.58	61.97	77.88	84.80	73.60	63.75	0.32	84.48	85.00	62.86	84.87	**85.17**
Agrw-IR-20	5.31	0.15	7.87	11.79	63.88	19.39	74.55	78.36	0.85	47.93	0.00	81.26	81.87	25.02	82.90	**83.00**
Agrw-IR-100	1.41	0.15	1.86	2.06	27.63	5.09	15.42	4.45	1.15	1.24	0.00	0.00	0.13	0.28	46.79	**66.02**
AssetNegotiation-IR-5	68.71	74.62	62.63	62.18	84.42	79.05	82.25	**85.16**	84.95	73.04	84.02	84.57	85.15	84.90	85.15	**85.16**
AssetNegotiation-IR-10	58.37	62.89	52.88	53.13	75.35	67.28	71.84	**76.32**	75.89	60.43	74.62	74.97	76.16	75.55	76.21	**76.32**
AssetNegotiation-IR-20	43.33	51.27	40.68	41.75	61.68	51.42	54.27	61.83	62.44	43.21	59.25	58.88	61.66	61.35	60.51	**62.56**
AssetNegotiation-IR-100	12.22	20.49	13.77	14.96	**24.78**	15.55	15.35	23.00	23.77	12.66	19.22	19.01	21.30	23.87	22.19	22.84
Hyperplane-IR-5	48.75	48.21	34.21	35.14	72.32	55.16	61.75	70.66	68.73	60.11	3.86	61.08	70.05	59.78	**81.66**	70.03
Hyperplane-IR-10	23.50	22.12	15.22	16.18	59.82	39.81	40.09	56.63	53.50	43.32	0.00	42.77	55.22	43.86	**71.45**	56.03
Hyperplane-IR-20	5.66	3.65	4.55	4.20	43.00	26.80	15.17	37.92	33.66	23.08	0.00	2.75	19.13	25.14	**55.98**	38.20
Hyperplane-IR-100	0.09	0.00	0.13	0.13	7.85	3.49	0.28	6.29	4.09	0.77	0.00	0.00	0.03	2.77	**16.50**	5.25
RBF-IR-5	20.95	18.47	18.15	18.54	79.71	84.39	62.67	87.84	**88.26**	71.89	0.03	85.50	86.74	86.86	80.01	86.69
RBF-IR-10	8.55	5.98	10.11	10.26	75.72	78.83	26.82	84.81	**85.32**	61.50	0.00	79.15	83.46	82.69	77.18	83.83
RBF-IR-20	4.25	0.39	5.09	4.51	67.31	71.49	11.81	80.46	**81.10**	45.72	0.00	30.03	68.19	77.17	70.60	79.19
RBF-IR-100	1.22	−0.05	0.57	0.47	41.70	38.89	3.84	**68.94**	67.60	1.92	0.00	0.00	2.91	61.84	54.80	63.57
SEA-IR-5	57.71	57.75	52.71	53.03	69.24	61.05	68.34	71.16	**71.34**	59.24	61.96	69.85	70.99	71.28	70.36	71.09
SEA-IR-10	43.65	44.89	36.33	36.79	55.87	47.74	54.11	58.07	**58.32**	44.35	39.77	54.62	57.93	**58.32**	57.41	58.09
SEA-IR-20	26.86	30.03	19.59	19.63	40.00	32.09	37.46	42.37	**42.62**	29.24	6.26	32.24	41.42	42.48	41.13	42.36
SEA-IR-100	1.55	2.60	1.94	2.10	12.00	8.36	6.82	13.08	**13.24**	7.00	0.00	3.71	10.45	13.06	12.39	13.09

Table 8 continued

Kappa	LNSE	DWM	DACC	ADACC	OCB	OBA	OBASHT	OBAD	LB	SAE2	AWE	AUE1	AUE2	ARF	HEB	KUE
Agrw-F1toF10-IR-1to20	48.80	54.29	45.39	45.82	64.06	59.58	66.49	76.17	69.47	63.47	66.45	69.60	**78.87**	68.70	62.35	77.81
AssetNeg-F1toF5-IR-1to10	80.94	81.27	76.63	76.35	85.64	81.76	87.16	**87.97**	87.93	77.66	86.50	85.50	87.84	87.69	86.54	87.56
RBF-drift-IR-1to10	16.88	15.00	17.69	17.73	**18.85**	17.63	15.36	17.32	16.75	15.91	11.45	13.58	15.63	18.25	14.81	18.36
SEA-F1toF4-IR-1to5	66.35	66.51	62.02	62.55	74.81	67.20	74.19	76.11	**77.60**	67.62	69.82	75.26	76.19	77.49	75.51	76.16
Agrw-F1toF10-IR-10to1	70.64	81.39	78.16	79.17	74.35	82.74	81.56	**88.59**	87.79	72.60	85.71	88.35	88.49	84.82	84.52	88.28
AssetNeg-F1toF5-IR-10to10	83.25	81.85	78.90	79.42	86.54	83.94	89.27	89.37	**89.68**	80.57	87.67	88.53	89.54	89.52	87.65	89.41
Poker-1-2vsAll	22.41	22.13	18.19	17.24	47.37	52.30	17.03	36.15	**72.78**	16.13	17.84	14.12	31.92	29.27	32.93	58.57
IntelLabSensors-1to9vsAll	0.00	97.95	91.46	84.32	99.82	99.82	99.82	99.82	99.82	81.62	0.01	**99.85**	0.01	99.82	98.82	**99.85**
IntelLabSensors-1to5vsAll	0.00	98.50	85.14	75.89	99.79	99.79	99.78	99.79	99.79	72.02	0.01	**99.83**	0.01	99.77	95.57	99.79
IntelLabSensors-1to3vsAll	0.00	95.80	81.14	67.96	99.46	**99.84**	99.75	99.75	99.75	66.29	0.01	**99.84**	0.01	99.74	95.30	99.75
IntelLabSensors-1vsAll	0.00	90.07	57.14	36.94	97.30	98.44	98.44	98.44	98.44	38.87	0.00	**98.86**	0.00	98.42	95.89	98.44
CovType-1-2vsAll	53.63	66.55	56.60	38.77	80.69	84.58	83.63	88.23	**89.92**	70.81	75.48	84.08	86.61	85.95	88.10	89.16
BNG_bridges-1vsAll	80.28	80.21	75.98	75.44	74.70	80.89	86.72	84.55	73.93	70.97	**87.51**	80.31	86.18	84.94	85.94	85.89
Avg. Kapppa	30.43	41.93	38.11	36.86	64.40	58.58	56.52	67.32	64.84	49.06	31.01	59.07	51.70	62.63	67.85	**69.84**

Bold values represent the best results

Table 9 Wilcoxon test for imbalanced data streams

Metric KUE versus	Accuracy *p* value	AUC *p* value	Kappa *p* value	G-Mean *p* value
LNSE	2.3E−10	5.6E−07	2.3E−10	8.2E−07
DWM	2.3E−10	2.3E−10	2.3E−10	2.3E−09
DACC	2.3E−10	5.6E−07	2.3E−10	1.7E−06
ADACC	2.3E−10	2.3E−10	2.3E−10	1.6E−08
OCB	2.2E−06	5.6E−07	6.9E−04	3.0E−02
OBA	2.0E−06	1.8E−06	1.5E−06	1.7E−04
OBASHT	6.2E−06	1.2E−01	2.0E−06	1.8E−06
OBAD	6.2E−01	5.8E−01	5.5E−01	7.6E−01
LB	2.8E−01	3.1E−03	3.4E−01	3.0E−01
SAE2	5.6E−07	5.6E−07	2.3E−10	2.3E−04
AWE	4.4E−09	1.5E−06	1.2E−09	1.2E−09
AUE1	2.1E−06	8.8E−06	2.1E−06	3.3E−06
AUE2	3.5E−03	2.4E−04	1.6E−04	1.3E−05
ARF	8.0E−04	1.7E−03	4.0E−04	1.4E−03
HEB	2.9E−03	2.3E−10	2.7E−03	1.2E−02

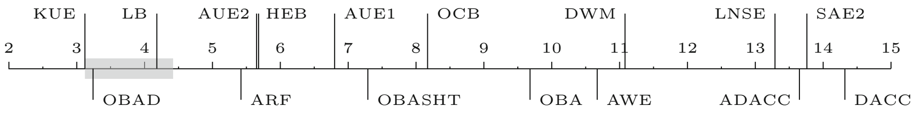

Fig. 8 Bonferroni–Dunn test for accuracy on imbalanced data

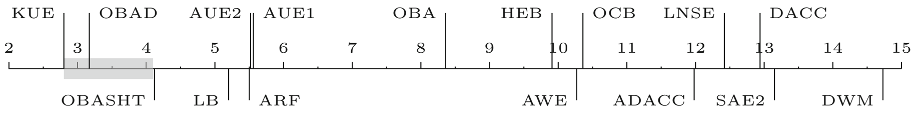

Fig. 9 Bonferroni–Dunn test for AUC on imbalanced data

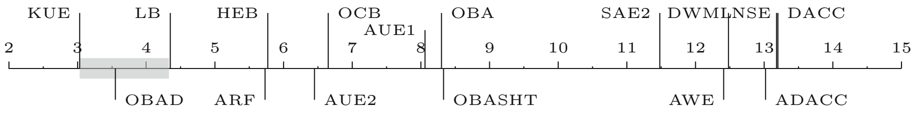

Fig. 10 Bonferroni–Dunn test for Kappa on imbalanced data

test. Figure 12 presents the pairwise comparison between KUE and reference methods with the respect to the number of wins, ties, and loses on all datasets. Figure 13 depicts the distribution of the frequencies of ranks achieved by all of the ensemble classifiers over all imbalanced datasets.

Finally, Figs. 14, 15 and 16 present detailed results over all processed instances in the stream for three selected datasets with respect to AUC, Kappa, chunk update time, and memory consumption.

The role of class imbalance in data stream mining While this work does not focus on imbalanced data stream mining, one must be aware that the issue of skewed class distributions may appear in any data stream problem. As instances arrive over time and we have no control over

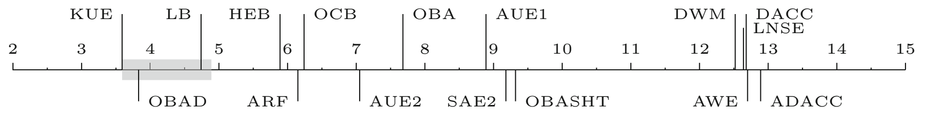

Fig. 11 Bonferroni–Dunn test for G-Mean on imbalanced data

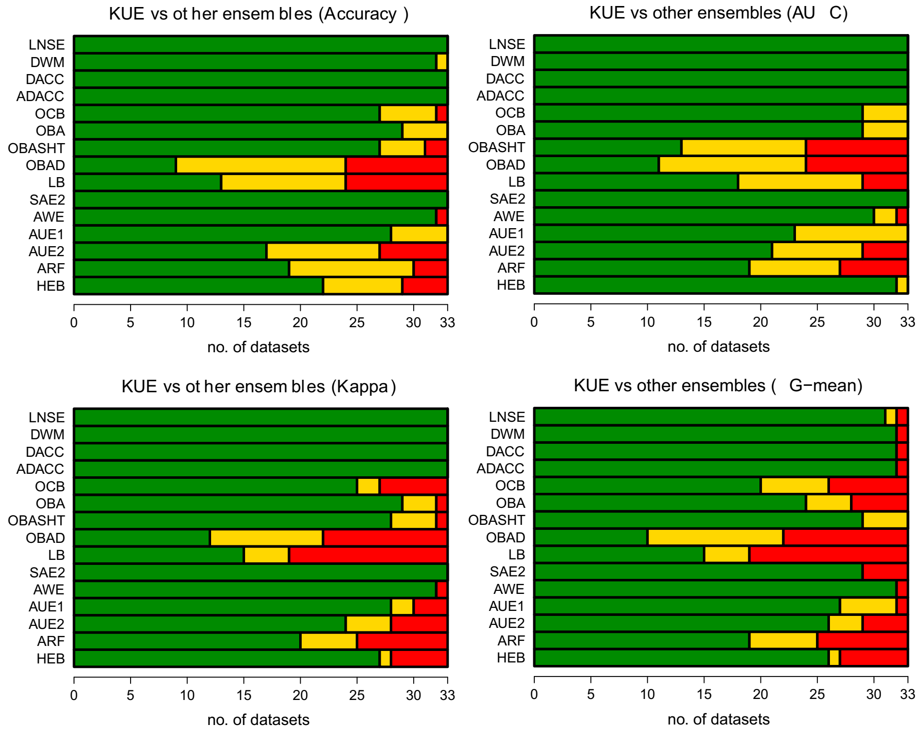

Fig. 12 Comparison of KUE and reference ensemble classifiers with respect to the number of wins (green), ties (yellow), and losses (red) over 33 imbalanced data stream benchmarks. A tie was considered, when the difference in obtained metric values were ≤ 0.05 (Color figure online)

the source of data, periodically we may obtain more instances from one of the classes. Such local class imbalance is usually not taken into account by the current solutions that either assume that the stream is always roughly balanced or that the class imbalance is embedded in the nature of the analyzed problem. However, such local distribution fluctuations may be harmful to a classifier and lead to creating a bias towards one of the classes that will propagate to new instances (even if they will not be imbalanced) and will be difficult to remove. As local imbalance will affect all the classifiers in the ensemble, one cannot deal with it by simply discarding one of the classifiers. Additionally, after a bias has been created, updating classifiers with new balanced distributions will not instantly remove it. Therefore, even shortly appearing imbalanced distributions may have long-term effects on any ensemble algorithm. That is why we postulate that even general-use data stream mining methods should display a high robustness to skewed class distributions.

Performance comparison using skew-insensitive metrics As accuracy is not a proper metric for evaluating imbalanced problems, we have selected three skew-insensitive prequential

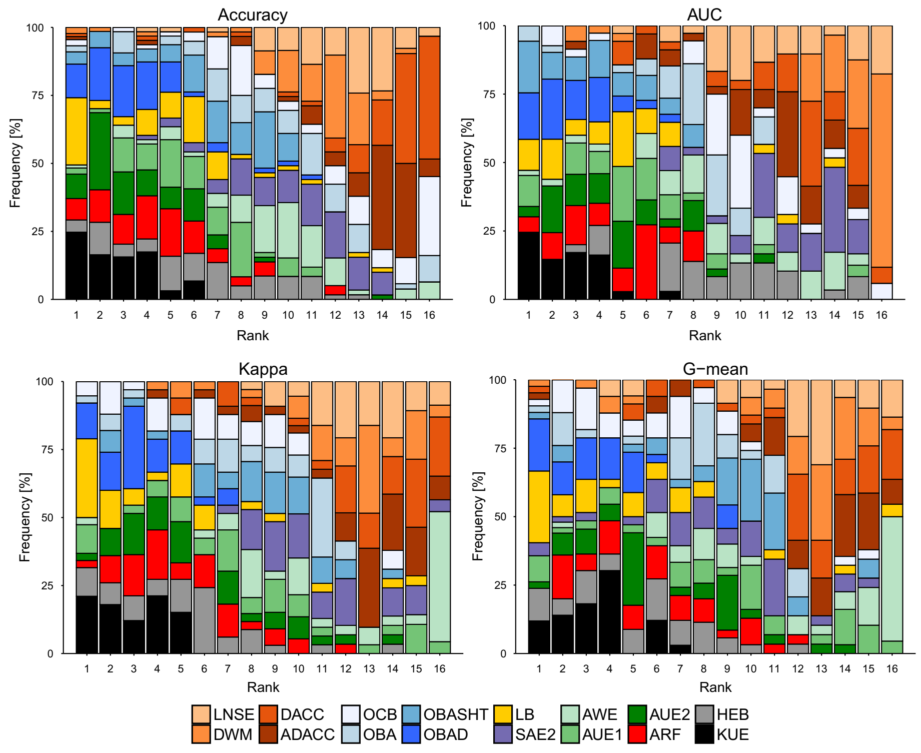

Fig. 13 Frequencies of ranks scored by ensemble classifiers on 33 imbalanced data stream benchmarks

measures: AUC, Kappa, and G-mean (Jeni et al. 2013; Brzeziński et al. 2018). For the purpose of evaluation, we have created 33 imbalanced benchmarks without and with concept drift, as well as with static or changing class imbalance ratios.

For the AUC metric, KUE offers the highest average performance, with OBASHT and OBAD following closely (87.86 vs. 87.39 and 87.37 respectively). This is confirmed by the rank test, where KUE scores 2.80, OBASHT 4.12 and OBAD 3.17. It is interesting to highlight that while the average AUC is higher for OBASHT, OBAD achieves a much better rank. This shows that OBASHT is not stable and displays a high variance among different datasets. On the contrary, KUE returns highly stable performance over all 33 benchmarks, always achieving both high AUC values and high positions in rankings. All other ensemble methods performed inferior to these three methods, showing that they are highly susceptible to skewed class distributions. It is interesting to see that AUE1 and AUE2, very well performing methods for standard data streams, return sub-par performance when handling skewed classes. Therefore, the only competitor for KUE is OBAD. While their AUC performance is similar, OBAD is a much slower and computationally expensive method than KUE, as proven by training time (0.4967 s vs. 0.0554 s) and memory consumption (5.43E−3 vs. 4.33E−4 RAM-Hours). This shows that when considering AUC as a metric, KUE offers an excellent performance both in predictive power and computational complexity.

For the Kappa metric, we achieve the highest differences between KUE and reference ensembles, which is to be expected as KUE optimizes this metric directly. Here, the three follow-up performers to KUE are OBAD, LB, and HEB (67.32, 64.84, and 67.85 vs. 69.84).

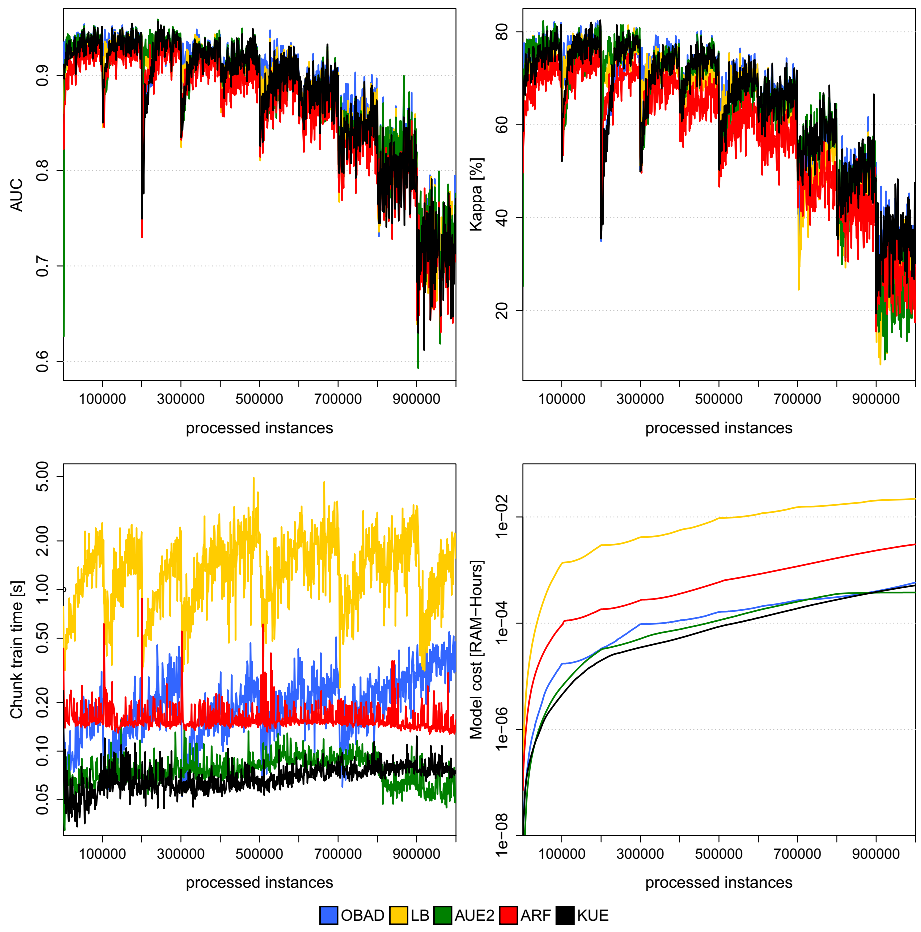

Fig. 14 Performance of top 5 ensemble methods according to their prequential AUC, prequential Kappa, chunk update time, and memory consumption on Hyperplane generator with imbalance ratio drift (1:1 to 1:20)

OBAD once again offers a stable performance when taking ranks into an account but at the cost of increased resource consumption.

Finally, for the G-mean metric, we observe a different behavior. KUE is still the best performing ensemble on average but OBASHT is not performing well for G-mean (with a rank of 9.32). The best performing reference methods are OCB, OBAD, and HEB, following closely the performance of KUE (75.89, 75.05, and 76.72 vs 77.18). However, this can be perceived differently, when analyzing the ranks of these algorithms (6.24, 3.83 and 5.89 vs 3.59). Once again, we can see that among the top-performing reference methods only OBAD offers a coherent performance between average G-mean and ranks but again at the cost of higher computational resource consumption than KUE.

To summarize, KUE offers the most stable performance on imbalanced data on all the three skew-insensitive metrics. This proves its increased robustness to imbalanced distributions as compared to reference ensemble methods. This is important from the perspective of potential periodical (or local) imbalance appearing in standard data streams, as one cannot anticipate

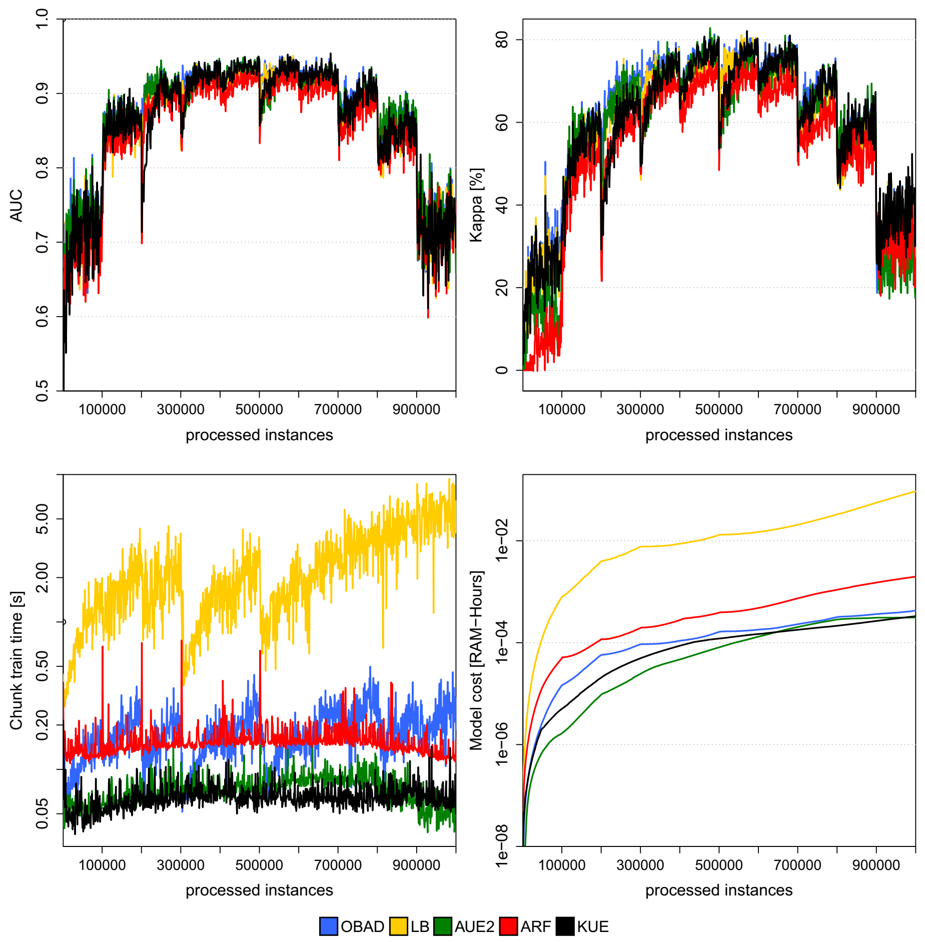

Fig. 15 Performance of top 5 ensemble methods according to their prequential AUC, prequential Kappa, chunk update time, and memory consumption on Hyperplane generator with imbalance ratio drift (1:10 to 1:1 to 10:1)

them and thus cannot employ efficiently dedicated algorithms to combat skewed classes. KUE combines excellent performance on standard data streams while being more robust to class imbalance, making it a highly attractive off-the-shelf algorithm for a diverse set of data stream problems.

Role of Kappa statistic KUE derives its good performance from the usage of Kappa statistic for both classifier selection and weighting. The advantage of Kappa lies in its applicability to both standard and imbalanced problems, as it can handle multi-class datasets and displays skew-insensitive characteristics. Therefore, contrary to other metrics, it can be seen as a more universal tool for monitoring the ensemble performance and a good choice when one requires an ensemble algorithm that can tackle a vast variety of data stream mining problems.

Role of feature subsets Another characteristic of KUE that improves its performance on imbalanced data streams is the usage of feature subsets for each base classifier. In the case

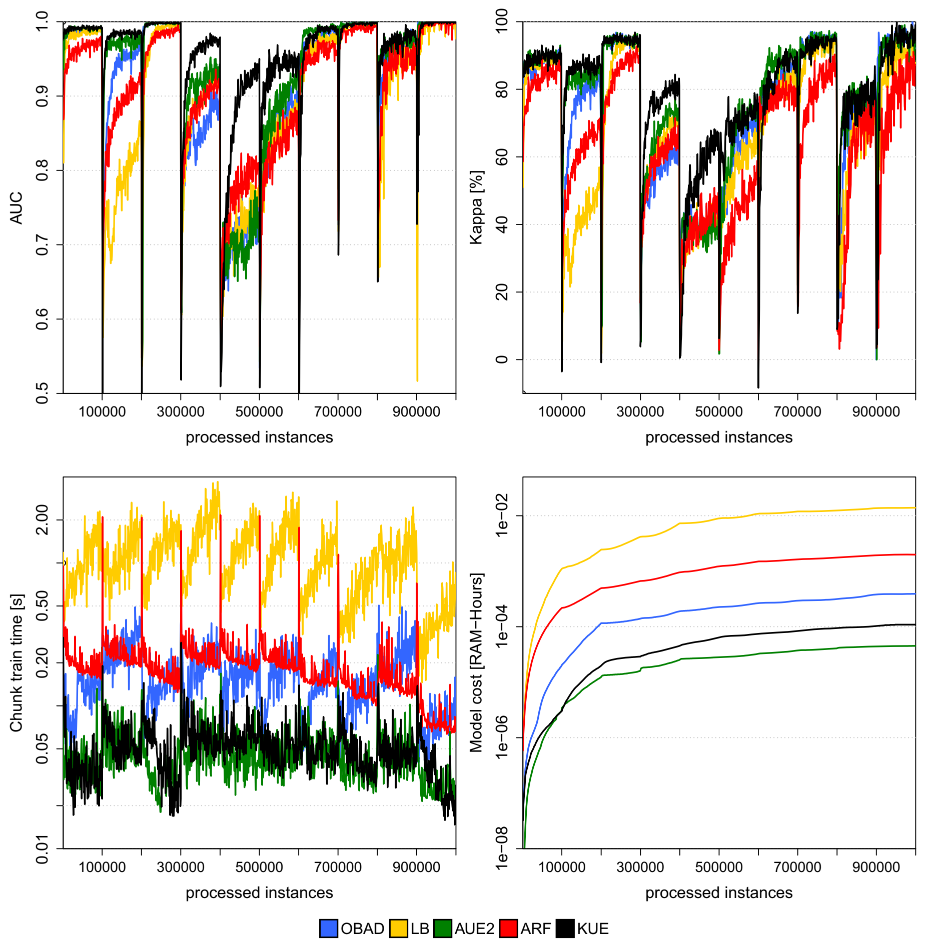

Fig. 16 Performance of top 5 ensemble methods according to their prequential AUC, prequential Kappa, chunk update time, and memory consumption on Agrawal generator with sudden concept drift and imbalance ratio drift (1:1 to 1:20)

of imbalanced problems, not only the imbalance ratio itself is a source of learning difficulty but also the instance-level characteristics. Even if the imbalance ratio is high but classes are easily separable, there will be no bias towards the majority class. The problem appears when instances are borderline or overlapping. These properties may be bounded with the features, as some of them will be characterized by lower or higher probability of correct separation. Therefore, base classifiers used in KUE have the possibility of discarding some of the more difficult features in used subspaces and thus reducing the bias towards the majority class. As our procedure for subspace creation is random, we cannot guide this as we would with a feature selection approach. At the same time, the random subspace creation does not impose additional computational costs on KUE, contrary to any feature selection. Therefore, using random feature subsets offers a good trade-off between improved robustness to skew-sensitive features and applicability to high-speed data streams.

4.5 Experiment 3: Analysis of Kappa Updated Ensemble properties

The third and final experiment aimed at investigating the specific properties of KUE and showcasing that our choice of its principles, components, and parameters is a valid one. We investigate the impact of the six most important aspects of the KUE algorithm on its performance: (1) the influence of Kappa vs accuracy to drive the ensemble components weighting and selection, (2) the contribution of the abstaining mechanism, (3) the contribution of the feature subspaces diversification, (4) the influence of the hybrid online architecture, (5) the number of classifiers that are trained on each new data chunk, and (6) the size of feature subsets that are used by each base classifier.

Influence of Kappa, abstaining, feature subspace diversification, and online architecture In order to evaluate the impact of the four discussed mechanisms on KUE performance, we have curated a set of experiments comparing different versions of KUE with one of the mechanisms being switched off. This allows us to compare their individual contributions to the KUE architecture. Figure 17 shows the comparison of performance among five different versions of KUE, using five data streams and three performance metrics.

From the results one can see that the discussed complete KUE architecture obtains the best performance, showcasing gains from embedding all four mechanisms into the learning process. There is not a single case when switching off any of the mechanisms would lead to an improvement in performance. There is a case for each mechanism showing that it contributes in a significant manner to overall KUE performance. Therefore, having all of them turned on allows for KUE to return excellent performance on a wide range of data stream problems. Thus, KUE can be seen as an off-the-shelf solution that could handle diverse classification problems without a need for any tedious parameter tuning or selecting which mechanisms should be switched off.

As for contributions of individual mechanisms, one can see that using online architecture leads to greatest improvements on all of metrics. This shows that combining block-based training of a new classifier with online updating of the ensemble members allows for better capturing both short-term and long-term changes, as well as adapting to local data characteristics without losing generalization capabilities. Diversity (i.e., using random feature subspaces of varying sizes) and Kappa-based weighting schemes are another big contributors, leading to better anticipation of drifts and faster recovery after change a (as seen on Aggrawal and Hyperplane datasets). Finally, abstaining is the least frequently used mechanism, but offers significant benefit to KUE in specific scenarios (as seen in Aggrawal datasets). Therefore, it protects KUE from relying its decision on non-competent classifiers in the ensemble, if such would ever appear.

Role of the number of base classifiers trained on each chunk We investigate if training more classifiers on each new chunk will lead to a predictive improvement in KUE. As we are using random feature subspaces for each classifier, the intuition dictates that such an approach should be fruitful. We examined the impact of training 1 (default parameter used in KUE) to 10 (equal to the full KUE pool) classifiers whenever new data becomes available. The trade-off between accuracy and computational cost averaged over all 93 used benchmarks is depicted in Fig. 18. Surprisingly, providing more classifiers for the KUE selection procedure does not lead to significant improvements in accuracy or Kappa for standard data streams, as regardless of the number of components the gains were statistically insignificant. At the same time, training each new classifiers and thus extending the KUE selection procedure leads to a significant increase in the update time for each batch. For imbalanced data streams

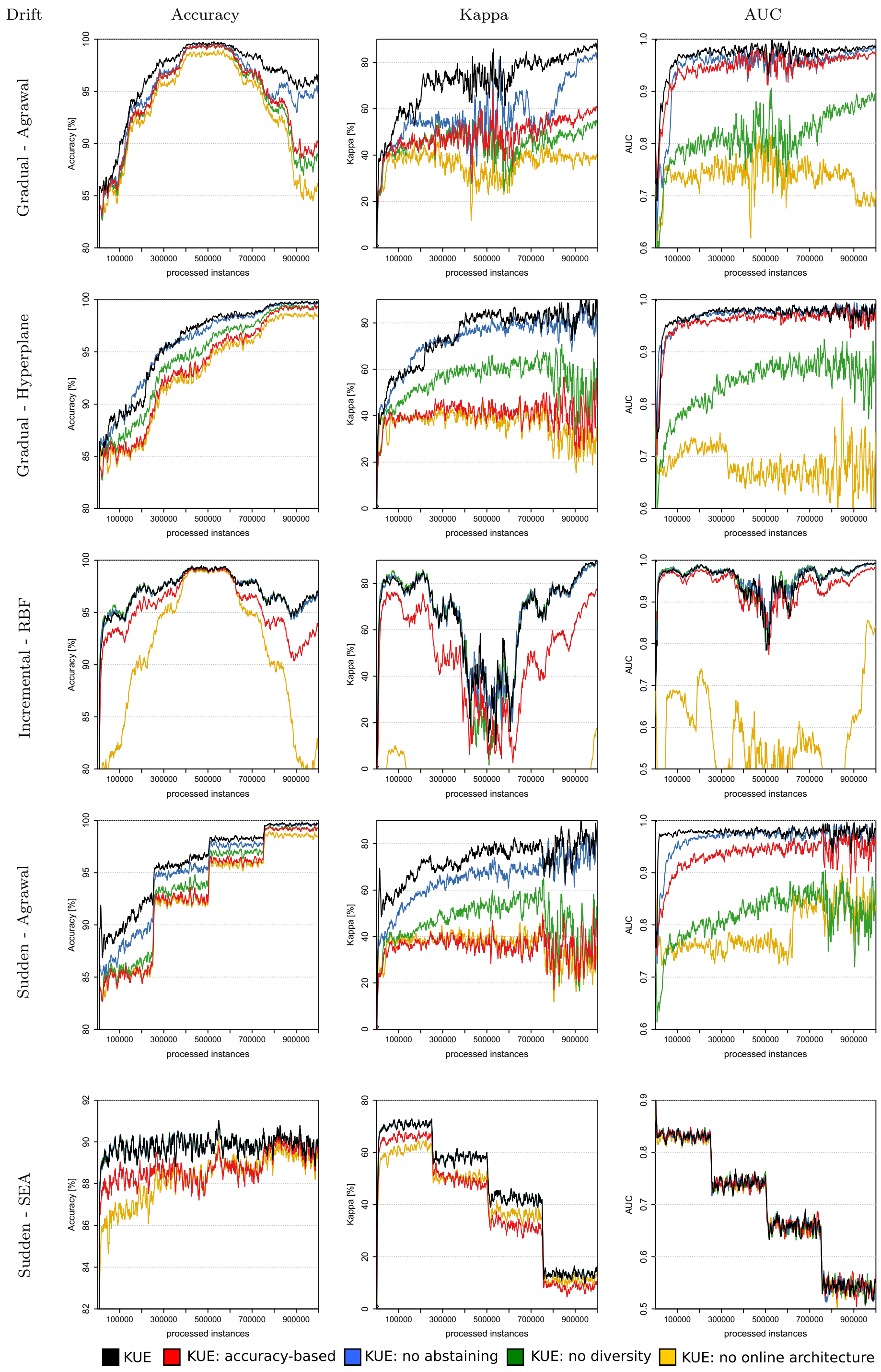

Fig. 17 Comparison between KUE and its individualized mechanisms on accuracy, Kappa, and AUC

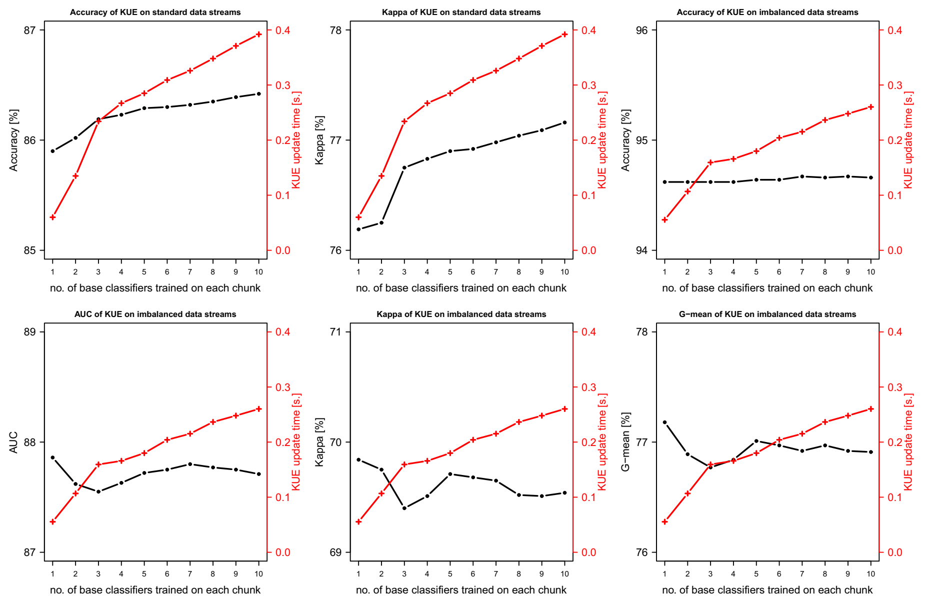

Fig. 18 Influence of the number of new classifiers trained for each new batch of instances on the KUE predictive power and update time

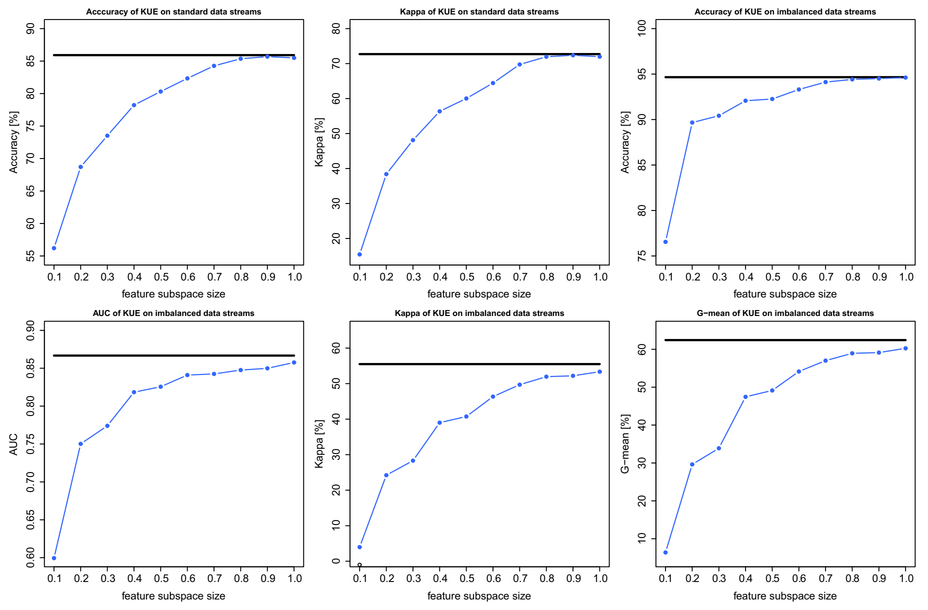

Fig. 19 Comparison between the proposed random feature subspace used in KUE (black line) and fixed size feature subspace (blue points), results averaged over all 93 data stream benchmarks (Color figure online)

and four different performance metrics used one can observe the same time dependencies as for standard streams. However, we observe even smaller, if any, gains in performance when more than a single classifier is trained per each chunk.

This may be explained by the fact that KUE does not forces addition of a new classifier to the ensemble for every chunk if it does not positively contribute to the ensemble. Therefore, with even a single classifier being trained, if the randomly selected features are of low quality, then it is not incorporated into the ensemble. At the same time, as each base classifier in KUE works in an online mode, each of them is updated with new instances, thus not losing the information coming from the new batch. This allows us to conclude that training a single new classifier on each batch of data leads to the best trade-off between predictive accuracy and required update time.

Impact of the feature sampling on the base classifiers We investigate if our proposed varying size of the feature subspace is better than a fixed size subspace. Figure 19 depicts the differences between the used sampling and fixed subspaces for each of metric and independently for standard and imbalanced data streams. We can see that the randomization in the size of feature subspaces always works better than using a fixed subspace size. We can explain this by the presence of feature drifts and the fact that the relevance of features evolves over time. Thus, a fixed feature subspace size is not capable of adapting to such dynamics, leading to either omitting important features (subspace too small) or incorporating too many redundant ones (subspace too big). KUE employs a classifier selection mechanism that adds a new base classifier to the ensemble only when it improves it. This indirectly alleviates the effects of incorrectly sampled feature subspace size, as such a classifier will be discarded. This can be seen as reducing the negative impact of variance in our randomized approach on the KUE performance.

5 Conclusions and future works

In this work, we have presented KUE, a new ensemble classification algorithm for drifting data streams. KUE offered a hybrid architecture, combining the advantages of online adaptation of base classifiers and block-based updating of the ensemble line-up. KUE used the Kappa statistic for simultaneous selection and weighting of base classifiers, which allowed to achieve a robust performance on standard and imbalanced data streams without the need for dedicated skew-insensitive mechanisms. KUE offered a better predictive power and adaptation to concept drift by training base classifiers on random subsets of features, which increased the diversity and capabilities for handling feature-based drifts. In order to reduce the impact of incompetent classifiers at a given state of the stream, KUE was empowered with an abstaining mechanism that removed selected classifiers from the voting procedure.

KUE was evaluated against 15 state-of-the-art ensemble algorithms on a wide set of 60 standard and 33 imbalanced data stream benchmarks. Such a wide-range study, backed-up with a statistical analysis of results, showed that KUE offers most the stable performance of all examined methods, regardless of data type and the metric used. Additionally, KUE was characterized by a low decision and update times, as well as memory consumption, making it a suitable choice for high-speed data stream mining. We showed an analysis of the KUE's main mechanisms and how the individually contribute to improving the predictive power.

We plan to continue our works on KUE and extend it to multi-label data streams, as well as to implement it on Apache Spark to learn from multiple parallel data streams in a distributed environment. Moreover, the exploration of the ROC metric for leading the selection, weighting of the classifiers, and heterogeneous ensemble schemes are promising lines for future research.

Acknowledgements This research was partially supported by the 2018 VCU Presidential Research Quest Fund and an Amazon AWS Machine Learning Research award.

References

Abdulsalam, H., Skillicorn, D. B., & Martin, P. (2011). Classification using streaming random forests. *IEEE Transactios on Knowledge and Data Engineering*, *23*(1), 22–36.

Almeida, P., Oliveira, L., de Souza, A., & Sabourin, R. (2016). Handling concept drifts using dynamic selection of classifiers. In *IEEE international conference on tools with artificial intelligence* (pp. 989–995).

Balle, B., Castro, J., & Gavaldà, R. (2014). Adaptively learning probabilistic deterministic automata from data streams. *Machine Learning*, *96*(1), 99–127.

Barddal, J. P., Enembreck, F., Gomes, H. M., Bifet, A., & Pfahringer, B. (2019a). Boosting decision stumps for dynamic feature selection on data streams. *Information Systems*, *83*, 13–29.

Barddal, J. P., Enembreck, F., Gomes, H. M., Bifet, A., & Pfahringer, B. (2019b). Merit-guided dynamic feature selection filter for data streams. *Expert Systems with Applications*, *116*, 227–242.

Barddal, J. P., Gomes, H. M., Enembreck, F., & Pfahringer, B. (2017). A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. *Journal of Systems and Software*, *127*, 278–294.

Barddal, J. P., Gomes, H. M., Enembreck, F., Pfahringer, B., & Bifet, A. (2016). On dynamic feature weighting for feature drifting data streams. In *European conference on machine learning* (pp. 129–144).

Barros, R. S. M., & Santos, S. G. T. C. (2018). A large-scale comparison of concept drift detectors. *Information Sciences*, *451*, 348–370.

Bertini, J. R., & Nicoletti, M. (2019). An iterative boosting-based ensemble for streaming data classification. *Information Fusion*, *45*, 66–78.

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

Bifet, A., Gavaldà, R., Holmes, G., & Pfahringer, B. (2018). *Data stream mining: With practical examples in MOA*. Cambridge: MIT Press.

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). MOA: Massive online analysis. *Journal of Machine Learning Research*, *11*, 1601–1604.

Bifet, A., Holmes, G., & Pfahringer, B. (2010). Leveraging bagging for evolving data streams. In *European conference on machine learning* (pp. 135–150).

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavaldà, R. (2009). New ensemble methods for evolving data streams. In *ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 139–148).

Błaszczyński, J., Stefanowski, J., & Zajac, M. (2009). Ensembles of abstaining classifiers based on rule sets. In *International symposium on methodologies for intelligent systems* (pp. 382–391).

Bonab, H. R., & Can, F. (2018). GOOWE: Geometrically optimum and online-weighted ensemble classifier for evolving data streams. *ACM Transactions on Knowledge Discovery from Data*, *12*(2), 25.

Brzeziński, D., & Stefanowski, J. (2011). Accuracy updated ensemble for data streams with concept drift. In *International conference on hybrid artificial intelligence systems* (pp. 155–163).

Brzeziński, D., & Stefanowski, J. (2013). Classifiers for concept-drifting data streams: Evaluating things that really matter. In *ECML PKDD workshop on real-world challenges for data stream mining* (pp. 10–14).

Brzeziński, D., & Stefanowski, J. (2014a). Combining block-based and online methods in learning ensembles from concept drifting data streams. *Information Sciences*, *265*, 50–67.

Brzeziński, D., & Stefanowski, J. (2014b). Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(1), 81–94.

Brzeziński, D., & Stefanowski, J. (2018). Ensemble classifiers for imbalanced and evolving data streams. *Data Mining in Time Series and Streaming Databases*, *83*(1), 44–68.

Brzeziński, D., Stefanowski, J., Susmaga, R., & Szczęch, I. (2018). Visual-based analysis of classification measures and their properties for class imbalanced problems. *Information Sciences*, *462*, 242–261.

Cano, A., & Krawczyk, B. (2018). Learning classification rules with differential evolution for high-speed data stream mining on GPUs. In *IEEE congress on evolutionary computation* (pp. 197–204).

Cano, A., & Krawczyk, B. (2019). Evolving rule-based classifiers with genetic programming on GPUs for drifting data streams. *Pattern Recognition*, *87*, 248–268.

Cano, A., Zafra, A., & Ventura, S. (2013). Weighted data gravitation classification for standard and imbalanced data. *IEEE Transactions on Cybernetics*, *43*(6), 1672–1687.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41*(3), 15.

Chen, S., & He, H. (2013). Nonstationary stream data learning with imbalanced class distribution. In H. He & Y. Ma (Eds.), *Imbalanced learning: Foundations, algorithms, and applications* (pp. 151–186).

Ditzler, G., Rosen, G., & Polikar, R. (2013). Discounted expert weighting for concept drift. In *IEEE symposium on computational intelligence in dynamic and uncertain environments* (pp. 61–67).

Dong, Y., & Japkowicz, N. (2018). Threaded ensembles of autoencoders for stream learning. *Computational Intelligence*, *34*(1), 261–281.

Elwell, R., & Polikar, R. (2011). Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, *22*(10), 1517–1531.

Faisal, M. A., Aung, Z., Williams, J. R., & Sanchez, A. (2015). Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: A feasibility study. *IEEE Systems Journal*, *9*(1), 31–44.

Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). *Learning from imbalanced data sets*. Berlin: Springer. https://doi.org/10.1007/978-3-319-98074-4.

Ferri, C., Hernández-Orallo, J., & Modroiu, R. (2009). An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, *30*(1), 27–38.

Gaber, M. M. (2012). Advances in data stream mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(1), 79–85.

Gama, J., & Castillo, G. (2006). Learning with local drift detection. In *Advanced data mining and applications* (pp. 42–55).

Gama, J., & Kosina, P. (2014). Recurrent concepts in data streams classification. *Knowledge and Information Systems*, *40*(3), 489–507.

Gama, J., Sebastião, R., & Rodrigues, P. P. (2013). On evaluating stream learning algorithms. *Machine Learning*, *90*(3), 317–346.

Gama, J., Žliobaite, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, *46*(4), 44:1–44:37.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, *180*(10), 204–2064.

García, S., & Herrera, F. (2008). An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. *Journal of Machine Learning Research*, *9*, 2677–2694.

Gomes, H. M., Barddal, J. P., Enembreck, F., & Bifet, A. (2017). A survey on ensemble learning for data stream classification. *ACM Computing Surveys*, *50*(2), 23.

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., & Enembreck, F. (2014). SAE2: Advances on the social adaptive ensemble classifier for data streams. In *ACM symposium on applied computing* (pp. 798–804).

Hoens, T. R., Polikar, R., & Chawla, N. V. (2012). Learning from streaming data with concept drift and imbalance: An overview. *Progress in Artificial Intelligence*, *1*(1), 89–101.

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. In *ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 97–106).

Jaber, G., Cornuéjols, A., & Tarroux, P. (2013). A new on-line learning method for coping with recurring concepts: The ADACC system. In *International conference on neural information processing* (pp. 595–604).

Jackowski, K. (2014). Fixed-size ensemble classifier system evolutionarily adapted to a recurring context with an unlimited pool of classifiers. *Pattern Analysis and Applications*, *17*(4), 709–724.

Jeni, L. A., Cohn, J. F., & De La Torre, F. (2013). Facing imbalanced data–recommendations for the use of performance metrics. In *Humaine association conference on affective computing and intelligent interaction* (pp. 245–251).

Kolter, J. Z., & Maloof, M. A. (2007). Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, *8*, 2755–2790.

Krawczyk, B. (2016). Learning from imbalanced data: Open challenges and future directions. *Progress in Artificial Intelligence*, *5*(4), 221–232.

Krawczyk, B. (2017). Active and adaptive ensemble learning for online activity recognition from data streams. *Knowledge-Based Systems*, *138*, 69–78.

Krawczyk, B., & Cano, A. (2018). Online ensemble learning with abstaining classifiers for drifting and noisy data streams. *Applied Soft Computing*, *68*, 677–692.

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., & Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion*, *37*, 132–156.

Kuncheva, L. I. (2013). Change detection in streaming multivariate data using likelihood detectors. *IEEE Transactions on Knowledge and Data Engineering*, *25*(5), 1175–1180.

Liu, A., Lu, J., Liu, F., & Zhang, G. (2018). Accumulating regional density dissimilarity for concept drift detection in data streams. *Pattern Recognition*, *76*, 256–272.

Marrón, D., Ayguadé, E., Herrero, J. R., Read, J., & Bifet, A. (2017). Low-latency multi-threaded ensemble learning for dynamic big data streams. In *IEEE international conference on big data* (pp. 223–232).

Matuszyk, P., & Spiliopoulou, M. (2017). Stream-based semi-supervised learning for recommender systems. *Machine Learning*, *106*(6), 771–798.

Mejri, D., Limam, M., & Weihs, C. (2018). A new dynamic weighted majority control chart for data streams. *Soft Computing*, *22*(2), 511–522.

Miller, Z., Dickinson, B., Deitrick, W., Hu, W., & Wang, A. H. (2014). Twitter spammer detection using data stream clustering. *Information Sciences*, *260*, 64–73.

Mimran, O., & Even, A. (2014). Data stream mining with multiple sliding windows for continuous prediction. In *European conference on information systems* (pp. 1–15).

Minku, L. L., White, A. P., & Yao, X. (2010). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, *22*(5), 730–742.

Morales, G. D. F., Bifet, A., Khan, L., Gama, J., & Fan, W. (2016). IoT big data stream mining. In *ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 2119–2120).

Nguyen, H. L., Woon, Y. K., Ng, W. K., & Wan, L. (2012). Heterogeneous ensemble for feature drifts in data streams. In *Pacific-Asia conference on knowledge discovery and data mining* (pp. 1–12).

Olorunnimbe, M. K., Viktor, H., & Paquet, E. (2018). Dynamic adaptation of online ensembles for drifting data streams. *Journal of Intelligent Information Systems*, *50*(2), 291–313.

Oza, N. C. (2005). Online bagging and boosting. In *IEEE international conference on systems, man and cybernetics* (pp. 2340–2345).

Pears, R., Sakthithasan, S., & Koh, Y. S. (2014). Detecting concept change in dynamic data streams. *Machine Learning*, *97*(3), 259–293.

Pelossof, R., Jones, M., Vovsha, I., & Rudin, C. (2009). Online coordinate boosting. In *IEEE international conference on computer vision* (pp. 1354–1361).

Pesaranghader, A., & Viktor, H. (2016). Fast hoeffding drift detection method for evolving data streams. In *European conference on machine learning and knowledge discovery in databases* (pp. 96–111).

Pesaranghader, A., Viktor, H., & Paquet, E. (2018). Reservoir of diverse adaptive learners and stacking fast hoeffding drift detection methods for evolving data streams. *Machine Learning*, *107*(11), 1711–1743.

Pietruczuk, L., Rutkowski, L., Jaworski, M., & Duda, P. (2017). How to adjust an ensemble size in stream data mining? *Information Sciences*, *381*, 46–54.

Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., Benítez, J., & Herrera, F. (2017). Nearest neighbor classification for high-speed big data streams using spark. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *47*(10), 2727–2739.

Ramirez-Gallego, S., Krawczyk, B., Garcia, S., Woźniak, M., & Herrera, F. (2017). A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, *239*, 39–57.

Ren, S., Liao, B., Zhu, W., & Li, K. (2018). Knowledge-maximized ensemble algorithm for different types of concept drift. *Information Sciences*, *430*, 261–281.

Ren, S., Liao, B., Zhu, W., Li, Z., Liu, W., & Li, K. (2018). The gradual resampling ensemble for mining imbalanced data streams with concept drift. *Neurocomputing*, *286*, 150–166.

Sethi, T. S., & Kantardzic, M. (2017). On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, *82*, 77–99.

Sobolewski, P., & Woźniak, M. (2013). Comparable study of statistical tests for virtual concept drift detection. In *International conference on computer recognition systems* (pp. 329–337).

Sobolewski, P., & Woźniak, M. (2017). SCR: Simulated concept recurrence—A non-supervised tool for dealing with shifting concept. *Expert Systems*, *34*(5), 1–12.

Srinivasan, A., & Bain, M. (2017). An empirical study of on-line models for relational data streams. *Machine Learning*, *106*(2), 243–276.

Sun, Y., Tang, K., Minku, L. L., Wang, S., & Yao, X. (2016). Online ensemble learning of data streams with gradually evolved classes. *IEEE Transactions on Knowledge and Data Engineering*, *28*(6), 1532–1545.

Triantafyllopoulos, D., Korvesis, P., Mporas, I., & Megalooikonomou, V. (2016). Real-time management of multimodal streaming data for monitoring of epileptic patients. *Journal of Medical Systems*, *40*(3), 45:1–45:11.

Van Rijn, J. N., Holmes, G., Pfahringer, B., & Vanschoren, J. (2018). The online performance estimation framework: Heterogeneous ensemble learning for data streams. *Machine Learning*, *107*(1), 149–176.

Vicente, R., Kinouchi, O., & Caticha, N. (1998). Statistical mechanics of online learning of drifting concepts: A variational approach. *Machine Learning*, *32*(2), 179–201.

Wang, B., & Pineau, J. (2016). Online bagging and boosting for imbalanced data streams. *IEEE Transactions on Knowledge and Data Engineering*, *28*(12), 3353–3366.

Wang, H., Fan, W., Yu, P. S., & Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. In *ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 226–235).

Wang, S., Minku, L. L., & Yao, X. (2015). Resampling-based ensemble methods for online class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering*, *27*(5), 1356–1368.

Wang, S., Minku, L. L., & Yao, X. (2018). A systematic study of online class imbalance learning with concept drift. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(10), 4802–4821.

Webb, G. I., Hyde, R., Cao, H., Nguyen, H. L., & Petitjean, F. (2016). Characterizing concept drift. *Data Mining and Knowledge Discovery*, *30*(4), 964–994.

Webb, G. I., Lee, L. K., Goethals, B., & Petitjean, F. (2018). Analyzing concept drift and shift from sample data. *Data Mining and Knowledge Discovery*, *32*(5), 1179–1199.

Woźniak, M., Kasprzak, A., & Cal, P. (2013). Weighted aging classifier ensemble for the incremental drifted data streams. In *International conference on flexible query answering systems* (pp. 579–588).

Woźniak, M., Ksieniewicz, P., Cyganek, B., & Walkowiak, K. (2016). Ensembles of heterogeneous concept drift detectors—Experimental study. In *Computer information systems and industrial management* (pp. 538–549).

Yuan, L., Pfahringer, B., & Barddal, J. P. (2018). Iterative subset selection for feature drifting data streams. In *33rd annual ACM symposium on applied computing* (pp. 510–517).

Zhai, T., Gao, Y., Wang, H., & Cao, L. (2017). Classification of high-dimensional evolving data streams via a resource-efficient online ensemble. *Data Mining and Knowledge Discovery*, *31*(5), 1242–1265.

Zhang, L., Lin, J., & Karim, R. (2017). Sliding window-based fault detection from high-dimensional data streams. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *47*(2), 289–303.

Zhang, Q., Zhang, P., Long, G., Ding, W., Zhang, C., & Wu, X. (2016). Online learning from trapezoidal data streams. *IEEE Transactions on Knowledge and Data Engineering*, *28*(10), 2709–2723.

Zhu, X., Zhang, P., Wu, X., He, D., Zhang, C., & Shi, Y. (2008). Cleansing noisy data streams. In *IEEE international conference on data mining* (pp. 1139–1144).

Žliobaite, I., Bifet, A., Read, J., Pfahringer, B., & Holmes, G. (2015). Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Machine Learning*, *98*(3), 455–482.

Žliobaite, I., Budka, M., & Stahl, F. T. (2015). Towards cost-sensitive adaptation: When is it worth updating your predictive model? *Neurocomputing*, *150*, 240–249.