Adaptive ensemble of self-adjusting nearest neighbor subspaces for multi-label drifting data streams

Gavin Alberghini [a], Sylvio Barbon Junior [b], Alberto Cano [a,*]

[a] Dept. of Computer Science, Virginia Commonwealth University, Richmond, Virginia, USA
[b] Dept. of Engineering and Architecture, University of Trieste, Italy

ARTICLE INFO

Article history:
Accepted 19 January 2022

Keywords:
Multi-label stream
Ensemble learning
Data stream
Concept drift

ABSTRACT

Multi-label data streams are sequences of multi-label instances arriving over time to a multi-label classifier. The properties of the stream may continuously change due to concept drift. Therefore, algorithms must constantly adapt to the new data distributions. In this paper we propose a novel ensemble method for multi-label drifting streams named Adaptive Ensemble of Self-Adjusting Nearest Neighbor Subspaces (AESAKNNS). It leverages a self-adjusting kNN as a base classifier with the advantages of ensembles to adapt to concept drift in the multi-label environment. To promote diverse knowledge within the ensemble, each base classifier is given a unique subset of features and samples to train on. These samples are distributed to classifiers in a probabilistic manner that follows a Poisson distribution as in online bagging. Accompanying these mechanisms, a collection of ADWIN detectors monitor each classifier for the occurrence of a concept drift on the subspace. Upon detection, the algorithm automatically trains additional classifiers in the background to attempt to capture new concepts on new subspaces of features. The dynamic classifier selection chooses the most accurate classifiers from the active and background ensembles to replace the current ensemble. Our experimental study compares the proposed approach with 30 other classifiers, including problem transformation, algorithm adaptation, kNNs, and ensembles on 30 diverse multi-label datasets and 12 performance metrics. Results, validated using non-parametric statistical analysis, support the better performance of the AESAKNNS and highlight the contribution of its components in improving the performance of the ensemble.

1. Introduction

Multi-label data stream mining merges two challenging tasks: multi-label classification and online learning. Multi-label classification is a generalization of the traditional classification task. In multi-label classification, each example's class contains multiple, non-exclusive labels instead of a singular class value. Ensembles are a widely popular method for the multi-label problem space [1–3], and have received significant research contributions over the last few years. The context of online learning, however, is actively being studied [4–6]. Data contained in real-world applications is appearing more frequently over infinite, continuous, time-evolving streams that present a unique set of difficult challenges for machine learning tasks. There are two main obstacles that contribute to the difficulty of data stream classification. These challenges include the requirement to utilize as few resources as possible (time and memory), and concurrently maintaining a high accuracy and robustness to concept drift. Online learning is based on the idea of real time feedback and learning instance by instance. This idea of how instances correlate to training and prediction time introduces the other issue associated with data stream mining called concept drift. Concept drift is the notion that overtime, data distribution for specific target attributes can change. Therefore, algorithms should adapt to changes in the data distribution dynamically. Ensembles are popular for adapting to concept drift since the algorithm may add/remove base classifiers to add/forget concepts appearing/fading in the stream. Many researchers have found that ensembles are highly competitive in the data stream mining context [7–10].

This paper presents a novel ensemble method for multi-label data stream classification named Adaptive Ensemble of Self-Adjusting Nearest Neighbor Subspaces (AESAKNNS). It proposes the idea of utilizing adaptive random subspaces to train base classifiers on unique and varying-size feature subspaces. Adaptive random subspaces increases the diversity of the ensemble and helps to adapt to concept drift by understanding the most relevant features at a given time. Moreover, it employs online bagging to pro-

* Corresponding author.
E-mail address: acano@vcu.edu (A. Cano).

vide different instance subsets to the base classifiers, further increasing their diversity. We leverage the MLSAkNN [11] as the base classifier due to its adaptive nature. When combined with feature and instance subspaces, a diverse set of self-adjusted MLSAkNNs demonstrate a robust performance and quick adaptation to concept drift. We also propose to monitor the concept drift on each of the feature subspaces. Upon warning of a drift, AESAKNNS starts learning a new ensemble in the background leveraging new, random feature subspaces. The dynamic classifier selection chooses the most accurate classifiers from the active and background ensembles to replace the current ensemble. AESAKNNS utilizes the advantages of the MLSAkNN as a multi-label base classifier, along with additional custom mechanisms for ensembles that handle concept drift and performance in online learning. The main contributions of this paper are as follows:

- AESAKNNS: an adaptive ensemble for multi-label drifting streams using nearest neighbors on random feature and instance subspaces.
- A methodology to increase the ensemble diversity by combining adaptive feature subspaces and online bagging.
- A self-adjusting nearest neighbor classifier as base model for the ensemble.
- A background ensemble to adapt to concept drift and new feature subspaces upon detection of a warning using ADWIN.
- A thorough experimental study comparing AESAKNNS to other state of the art models, including an in depth analysis of the ensemble mechanisms and how they contribute to improve the classification performance.

2. Multi-label stream classification

2.1. Multi-label data

A multi-label data stream is defined as a potentially unbounded sequence $< S_1, S_2, \ldots, S_n, \ldots >$, in which each element S_j is a collection of instances (batch scenario) or a single instance (online scenario). Each instance is defined as (x, y) where x represents the instance features and y represents a set of labels simultaneously associated with the instance. These labels are sometimes represented as a generic interpretation of multi-class data. Just as these problem spaces are related, common methods in the literature for multi-label learning involve either bringing a model from the multi-class context into the multi-label context (Algorithm Adaptation), or transforming data into problems that can be solved via more traditional methods (Problem Transformation). Problem transformation focuses on the idea of changing multi-label examples into problems that can be solved using already established methods. Label combination (LC), also known as label power set (LP), methods transform multi-dimensional label sets (y) into a single class value towards converting the multi-label problem into a multi-class problem. Some known issues with this method include over training and its worst case computational complexity. Binary Relevance (BR) is a problem transformation method that decomposes a d-dimensional label vector into d number of binary classification problems. This allows standard binary classifiers to be used and predict the possibility of each label. There are some issues raised by this strategy [12] stemming from the loss of any label correlation when the problem is decomposed. Though there are multiple mechanisms to combat the issue of lost label correlations in literature [12]. Classifier Chains (CC) are a common means to rectify the loss of label correlations associated with methods such as binary relevance. The idea behind this method is to supply label predictions as features to classifiers further down the chain. If a correlation between labels exists between the provided labels and the label being actively predicted, this method will detect it. Unfortunately, this means that label correlation discovery is based on the order of predicted labels in the classifier chain. Commonly, several variations of the label order are trained at the same time and the highest achieving CC is chosen. Algorithm adaption methods take the opposite approach and focus on making changes to decision functions. This allows a previously single class model to now operate in the multi-label context, including methods for feature selection [13]. One example algorithm adaption includes MLkNN [14]. The MLkNN algorithm developed by Zhang et. al. performs this adaptation by combining the kNN approach with Bayesian probabilities. The algorithm begins by calculating the nearest neighbors of a given training instance. Then, using the gathered neighbors, a membership counting vector and category vector can be derived for label sets. Thus, the prediction requires the prior and posterior probabilities, as well as the neighbors themselves, which are all directly calculated from the training data.

2.2. Data streams

As mentioned prior, the online learning scenario deals with instances arriving to a model over time. In this case, algorithms must be able to not only learn from the data it has seen, but be able to adjust to new knowledge as it arrives. A key data consideration in this context includes Oza et al. [15] and their work with boosting and bagging methods for the online problem space. In their work it is shown that the binomial distribution of instances in batched based bagging can be applied to the online scenario via a Poisson (1) distribution. To carryout the online sampling, inbound instances are learned k number of times where k is set according to Poisson(1). This work has enabled many algorithms, especially lazy learners, to make the leap from batch to online learning. Concept drift is the term used to describe how the statistical properties of target label set y change over time. Formally we define concept drift as $P_t(x, y) \neq P_{t+\Delta}(x, y)$ where $P(x, y)$ represents the joint distribution between features and labels at time t. Some of the different types of concept drift are detailed below:

- Sudden concept drift: The scenario in which there is an instant change in the underlying data distribution at a particular time t. Models built prior to the drift are immediately unreliable and should be discarded.
- Incremental concept drift: The situation where there is steady progression through multiple concepts over a time interval (t_1, t_2). Each subsequent shift results in a different concept from the original data distribution and closer to the target distribution. Models incrementally adapt to the drift.
- Gradual concept drift: The context in which incoming data is alternating between two different concepts with a growing bias toward the new data distribution over time. Models can gradually adapt to the drift.
- Recurring concept drift: The idea that a previously seen concept can potentially reappear once or multiple times in the future [16]. Models can be saved and restored when the recurring drift reappears.

While all of the listed forms of concept drift focus on when and how the drift appears, another important factor is to consider if drift is contained within one concept. Real concept drift describes a change that invalidates prior decision boundary knowledge of a class. This would invalidate any prior knowledge of the concept and require new knowledge to supplement the shift. Conversely, virtual concept drift is a change that only affects the distribution of data within a known concept but not the decision boundary between them. Being able to differentiate between these two types of drift helps avoid unnecessary changes to the current knowledge.

In [17], a feature selection scheme is developed specifically for the multi-label streaming problem space. It leverages correlation degrees, relevance analysis, and redundancy analysis to derive the best feature sets for correlated labels. Another issue associated with data streams is that the proportion in which each class is expressed varies throughout the duration of the stream. Many supervised learning algorithms naively rely on the assumption of equal class distributions. When data proportions are skewed, one can expect to see poor predictive performance for minority classes. In order to account for this difference in data, it becomes necessary to adapt the learning. One such mechanism of adaption includes random sampling methods such as over-sampling and under-sampling [18,19]. The work of Tarekegn et al. [20] explores several additional methods for combating the issue of class imbalance. These methods include classifier adaption, ensemble methods, and cost-sensitive. All of which apply some type of augmented sampling or meta evaluations to dynamically adjust with the data. As the additional dimension of multi-label data is applied to the problem space, the need for more comprehensive solutions arise.

2.3. Nearest neighbors for multi-label data streams

For our paper, we look at several additional adaptations of the kNN algorithm to the multi-label data stream problem. In the work by Roseberry et al. [21] they proposed MLSAMkNN, a multi-label kNN algorithm that leverages majority voting along among individual labels with a self-adjusting short and long term memory structure (STM and LTM). The key benefits of this system's memory mechanism is twofold. The prior being that the short term memory has the ability to adapt to rapid changes in concept. The latter being that reoccurring patterns in data can be stored in the long term memory and recalled as needed. In order to insure the STM and LTM do not contradict each other, a cleaning procedure is defined. This cleaning procedure leverages a threshold for each label that compares a set of instances A to the current content of the STM. If there are a number of inconsistent labels observed, then those in conflict within the LTM are removed. Conversely, when the STM is shrunk, the excess data is cleaned and transferred to the LTM, at which point it is continually cleaned as new instances fill the STM. One drawback of this design is the necessity to keep memory consumption low. To alleviate this issue, the authors implemented a kmeans++ compression algorithm to reduce the amount of data in the LTM as it becomes necessary. In their following work [22], they propose MLSAMPkNN. This algorithm leverages an adaptive window akin to the short term memory of MLSAMkNN as well as a punitive system for removing erroneous data. Traditionally, a fixed size sliding window allows old instances to be removed and new instances to enter at a fixed rate. This allows for gradual shifts in concept to be captured as data slowly adjusts within the window. However, the authors of this paper implement a sliding window of flexible size. The advantage of this flexibility is that the window can be evaluated at different intervals and resized accordingly to match abrupt concept drift. In addition to the window, a punitive system is used to identify and remove potentially erroneous instances. After a prediction is made for an incoming instance, the labels for each of the nearest neighbors is checked against the true label set of the inbound instance. It a neighbor provides a label that is not identified as the true label of the inbound instance, an error is accrued. This mechanism helps to constrain memory and computational requirements by eliminating instances from the window. It also complements the nature of the self adjusting window by removing older instances as they begin to produce errors due to drift. Finally, in [11], the authors proposed MLSAkNN as a direct improvement of the MLSAMPkNN algorithm. The focus of MLSAkNN is to provide self adapting configurations to the multi-label problem space. It does so through two major methods, the first being a dynamic k value that self-adjusts over time for each label, adapting automatically to the best parameter settings in real-time. This is carried out by continually evaluating different ks for each label. After the currently selected k is used to evaluate an instance, it is then compared against other k values. The second mechanism is an adaptation of the MLSAMPkNN punitive for use within each label space. In the updated version each instance now has a label mask that informs the algorithm if the instance should be included in the neighbor calculation for a particular label.

2.4. Ensemble learning

Ensemble learning is a popular methodology of classification in machine learning. Ensembles are collections of classifiers that represent diverse knowledge on the problem space. These classifiers work together in order to create combined knowledge scenarios where predictions are more accurate and robust. This learning method requires that both the classifiers within the ensemble contain novel information about the problem space and that they are powerful enough to contribute to predictions in a positive way. In this paper, we are focused on two topic areas of ensemble learning: (i) ensembles that work on real time drifting data streams, and (ii): ensembles for multi-label data and their applicability for streaming settings.

2.4.1. Ensembles for data streams

Over recent years of research, ensemble learning has been identified as an effective method for data stream mining. Adaptive Random Forests [5] is a traditional ensemble method with high popularity. This algorithm focuses on adaptive example resampling to generate forests in the online setting. Tree algorithms traditionally rely on the ability to repeatedly scan a static dataset to perform split decisions. In the online setting the data is dynamic, which requires split decisions to come from real time metrics. The Adaptive Random Forests algorithm leverages Hoeffding Trees to overcome this limitation. Also leveraging the use of drift detection methods in order to replace the existing classifiers as concepts change. Some advantages of this model include the ability to build trees in parallel without sacrificing predictive performance. As well as maintaining a decoupled relationship to aggregation and concept drift detection.

In the work of Sun et al. [10] ensembles were leveraged to allow for tracking of multiple concepts present in the data. Each component classifier utilized a one-versus-all approach in order to associate examples with a particular class. Each class based model is implemented as a binary classifier that is able to output the posterior probability, which can be represented using Baysean theory. When training, the models update their posterior probabilities with each inbound instance. The authors include dynamic sampling in order to represent the minority classes among the shifting data. Throughout the data stream, the ensemble identifies three possible concept states: the emergence of a class, the disappearance of a class, and the emergence of a previously disappearance class. The benefit of this ensemble mechanism is the adaption to concept drift through consideration of when to apply knowledge.

Museba et al. [23] recently proposed an adaptive ensemble for non-stationary data streams. Their algorithm maintains a collection of classifiers that are evaluated on accuracy and diversity. As the ensemble is introduced to new concepts, new classifiers are created in order to capture the knowledge. Passive methods to limit the ensemble size are introduced, including hyper parameters that define the criteria to remove a classifier from the collection. On data drift detection, all of the learners in the collection are reset. These modules enable the algorithm to adapt to the non-stationary data over time, improving performance in this context.

Another algorithm that leverages meta-analysis is the algorithm proposed in Cano et al. [7,8]. The ensemble mechanisms proposed highlight how diversity, while beneficial to data streaming problems, does not translate directly to boosts in performance. In order to effectively utilize different component classifiers, an adaptive confidence threshold was established. This threshold allowed for shifting criteria definitions for the participation of component classifiers based on the state of the data. Specifically, as the ensemble predicts correctly, the threshold is relaxed and additional classifiers can participate in the voting. Consequently, as incorrect predictions happen the threshold is increased and only more confident predictions are allowed to influence the voting. An inherent draw back of their design is that highly confident, but incorrect classifications carry heavy weight in driving the system. However, diversity within ensembles helps to mitigate overconfident and inaccurate predictions. The important aspect of online learning algorithms is the ability to adapt to the evolving data stream. Ensemble methods allow for meta evaluations of several classifiers to detect changes in performance and predict the arrival of concept drift. Through the application of boosting methods new component classifiers capture additional knowledge from new concepts through training on the drifted data. These reasons explain why ensemble methods perform better in the data stream mining context, because they are able to adjust to non-stationary data dynamically.

2.4.2. Ensembles for multi-label data

There are two major avenues for dealing with multi-label data: problem transformation and algorithm adaptation. Ensembles have been widely adopted in recent years as a means for implementing problem transformation methods [24–27]. When used for binary relevance, ensembles are leveraged to provide an independent classifier per label. The advantage of this approach is that any binary classification algorithm is usable in this context. In the LP problem domain, ensembles are used to distribute a large multi-class representation among multiple classifiers. This allows for label dependencies to remain intact, however, data in this problem space only represents a fraction of the total label combination power set. Without mitigation strategies, the large set of possible classes can be detrimental to predictive and computing performance.

One such strategy is proposed in Gatto et al. [28], where Jaccard indexes are used to map the label space into hybridized partitions. This helps to combine the benefits of binary and label power set methods by allowing label dissimilarity to drive the decision on assuming label dependencies. In a similar study [29] combinatorial optimization is utilized to determine the optimum k-subsets of labels to utilize against LP classifiers. Two version of this algorithm were proposed, one with disjoint label sets and another with overlapping. By reducing the number of label combinations to consider the time and memory performance of classification algorithms can improve. Additionally, if label set predictions can accurately determine dependent labels, the performance of dependency seeking algorithms can increase as well.

Classifier chain ensembles express binary classification with different label chain orders to improve the chance of discovering the true label dependency. In the work of Senge et al. [30] some of the core obstacles of classifier chains are considered. Classifier chains are particularly susceptible to attribute noise when trained on true label values. The authors propose to alternatively train the model on predicted labels instead of true values to alleviate this issue. They also pursue a nested stacking approach where several meta classifiers learn on the combined outputs of binary predictions for a subset of labels.

In contrast to the problem translation use case, ensembles are also popular for algorithm adaption [2,3]. They provide similar benefits in the multi-label context as seen for traditional classification problems. In the multi-label problem space, ensembles are often effective measures to mitigate class imbalance. If even a single label experiences severe class imbalance, it must be addressed or the multi-label prediction will fail. There are several recent papers in both the research and practical fields that deal with class imbalance for multi-label data [31].

In the work of Zhang et al. [31], the authors drive to design a learning strategy that explores both label correlations and class imbalance simultaneously. Each class label utilizes a binary learner along with a set of K coupling multi-class learners. The multi-class learners attempt to discover correlations among K number of label pairs while the binary classifier helps mitigate class imbalance. A predictive model for each label is then aggregated from the confidences of the induced classifiers. Finally, the generated models are leveraged for the final prediction. The benefit of this approach is the mitigation of class imbalance while still leveraging binary and multi-class problem spaces.

In the work of Wu et al. [32], they propose a multi-label tree ensemble algorithm that is meant to exploit the dependencies between labels by learning them as hierarchical trees that reflect the intrinsic label dependency of the data. Each internal node looks to partition the data into smaller subspaces of similarly labeled instances. The labels used to partition the data are then passed to children nodes that further examine the label space. While each tree may be able to partition the entire dataset, over fitting is a likely scenario. The authors then propose to combine multiple trees into an ensemble in order to reduce the risk of overfitting by providing trees with different data. This also improves the scaleability of their algorithm. In the work of Wei et al. [33] a probabilistic label tree is proposed for the multi-label streaming context. The algorithm leverages both tree structures and binary classification nodes to interpret data that is broken up by partially observed labels. They also implement update procedures that will detect new labels by passing instances down the full tree and updating the corresponding binary classifiers.

The ideas proposed by Huang et al. [34] include methods for combating common issues with problem transformation methods. They propose a method of learning unique features based on each class label. Their work reached the same conclusion as Zhang and Li [35] and demonstrated that individual labels have their own representative features. This idea helps to guide meta-based ensemble methods to better understand relevant features for individual labels in data.

2.4.3. Ensembles for multi-label data streams

When dealing with multi-label data streams, we must understand the issues of both multi-label data and data streams. Detailed in the work of Zheng et al. [36], class imbalance is a significant problem for data streams. They discuss how existing methods for combating class imbalance rely on static proportions of imbalance over time, which is unrealistic for practical applications. The important consideration being that throughout a stream data proportions are constantly shifting. To compensate for the dynamic problems of concept drift, classifiers should ideally be able to incorporate mechanisms to handle changing class representations together with concept drift.

The continued work of Zhang et al. [37] depicts a re-sampling ensemble framework to combat this issue. Their method depicts both static and dynamic classifiers that learn from the data using circular caches of instances. Once the cache of instances is filled from the data stream a new block of data is recognized and a corresponding dynamic classifier is created. This classifier iterates through the training data while calculating reinforcement weights for minority classes and recent instances. Additionally the algorithm maintains a resampling buffer to maintain relevant instances

to use for sampling procedures. This helps to keep old concepts from creeping into the active ensemble through resampling.

Sun et al. [38] establish in their work an ensemble algorithm supported by Jensen-Shannon concept drift detection. This detection method uses a comparison measure between two windows of instances to detect changes in the underlying data distribution. Their algorithm also leverages infrequent label pruning as a method to improve classification performance by exploiting highly observed label dependencies. This can be beneficial to computing resources by limiting the potential problem space. However, minority classes are possibly under represented or forfeit entirely based on the pruning method.

In another study, Sousa et al. [39] reveal an experimental analysis of two rule-based algorithms that leverage label subset rules to form decision boundaries. These rules increase the performance of the algorithm by limiting the problem space of each prediction and adapting to changes in the data. The novelty for this algorithm is in the fact rules are generated on subsets of labels instead of a single or all labels, again exploiting potential label dependencies.

As an alternate approach to supervised learning, Costa et al. [40] purpose an unsupervised algorithm for multi-label classification streams that utilizes an offline and online strategy. Their method applies runs k-means clustering to identify the possible labels, then the euclidean distance of inbound instances are used to derive class predictions and discover new concepts. In their later work, [41] they apply a pruned set transformation method to their algorithm. The pruned sets increase correlation of the existing concepts and aides the computational complexity of their algorithm. Concurrently, the authors looked to develop a new method for novelty detection [42] through the MINAS-BR algorithm. The novelty detection algorithm creates and evaluates micro-clusters as inbound data falls outside of existing concepts. A euclidean distance is measured from the micro cluster to the nearest known concept. If the distance is greater than the standard deviation of distances between examples and the centroid, then a new concept is created.

In the work of Cerri et al. [43], an unsupervised method for multi-label stream classification is proposed. The algorithm uses both an offline and online component, consisting of a number of self organizing maps (SOMs) in the offline phase, and kNN based selection of neurons in the online phase. The online portion of the algorithm is configured to update the SOMs as new instances arrive off the stream. The authors continue their work in [44] where they add additional mechanism for concept drift and utilize Bayesian probability rules to better control the adaptability of the algorithm. It is worth noting that the required offline training to generate SOMs is a drawback when comparing this to other algorithms.

Our proposed algorithm AESAKNNS is designed to meet the challenges of this problem domain via adaptive, multi-label component classifiers and intelligently designed ensemble methods.

3. Adaptive Ensemble of Self-Adjusting Nearest Neighbor Subspaces

This section introduces the Adaptive Ensemble of Self-Adjusting Nearest Neighbor Subspaces (AESAKNNS) as a front running algorithm for multi-label data stream mining. The pseudo-code is presented in Alg. 1 and the flowchart is depicted in Fig. 1. We will describe the motivation, architecture, and components of the ensemble in the following.

Algorithm 1: Adaptive Ensemble of Self-Adjusting Nearest Neighbor Subspaces.

Input:
S: data stream
m: number of base classifiers (ensemble)
w: window size
Symbols:
E: ensemble of m γ classifiers
E': background ensemble of m γ' classifiers
α: ADWIN detector for each m classifier
τ: feature subspace for each m classifier
h: Hamming score for each m classifier
s: subset accuracy for each m classifier
Z: label set prediction of a base classifier

for $S_i \in \{S_1, ..., S_n\}$ **do**
  if $\mathcal{S}_1$ **then** ▷ *Ensemble initialization*
    for $j \in \{1, ..., m\}$ **do**
      $h_j \leftarrow$ Hamming score of γ_j at instance S_1
      $s_j \leftarrow$ subset accuracy of γ_j at instance S_1
      $\rho_j \leftarrow$ random subspace size
      $\tau_j \leftarrow$ r-dimensional set of features
      $\gamma_j \leftarrow$ new MLSAkNN on $\tau_j(S_1)$
    end
  end
  $Z \leftarrow E$ prediction on S_i
  $\alpha \leftarrow$ ADWIN update on S_i and Z
  if α *warning for any* $\gamma \in E$ **then** ▷ *Concept drift detected*
    for $j \in \{1, ..., m\}$ **do**
      $\gamma_j \leftarrow$ reset learning
      $\gamma_j \leftarrow$ incremental train of γ_j on $\tau_j(S_i)$
      $\alpha_j \leftarrow$ new ADWIN
    end
  end
  for $j \in \{1, ..., m\}$ **do** ▷ *Ensemble update*
    $k \leftarrow Poisson(1)$ value for weighting
    if $k > 0$ **then**
      $S'_i \leftarrow k$-instance weighting of S_i
      $\gamma_j \leftarrow$ incremental train of γ_j on $\tau_j(S'_i)$
    end
  end
  if α *warning was detected* **then**
    if $E' = \emptyset$ **then** ▷ *Background ensemble initialization*
      for $j \in \{1, ..., m\}$ **do**
        $\rho'_j \leftarrow$ random subspace size
        $\tau'_j \leftarrow$ r-dimensional set of features
        $\gamma'_j \leftarrow$ new MLSAkNN on $\tau'_j(S_i)$
      end
    end
    for $j \in \{1, ..., m\}$ **do** ▷ *Background ensemble update*
      $h'_j \leftarrow$ Hamming score of γ'_j at instance S_i
      $s'_j \leftarrow$ subset accuracy of γ'_j at instance S_i
      $k \leftarrow Poisson(1)$ value for weighting
      if $k > 0$ **then**
        $S'_i \leftarrow k$-instance weighting of S_i
        $\gamma'_j \leftarrow$ incremental train of γ'_j on $\tau'_j(S'_i)$
      end
    end
    if i - α *warning timestamp* $= w$ **then** ▷ *Replace base classifiers*
      $E \leftarrow Select(E \cup E', h \cdot s, h' \cdot s', m)$
      $E' \leftarrow \emptyset$
    end
  end
end

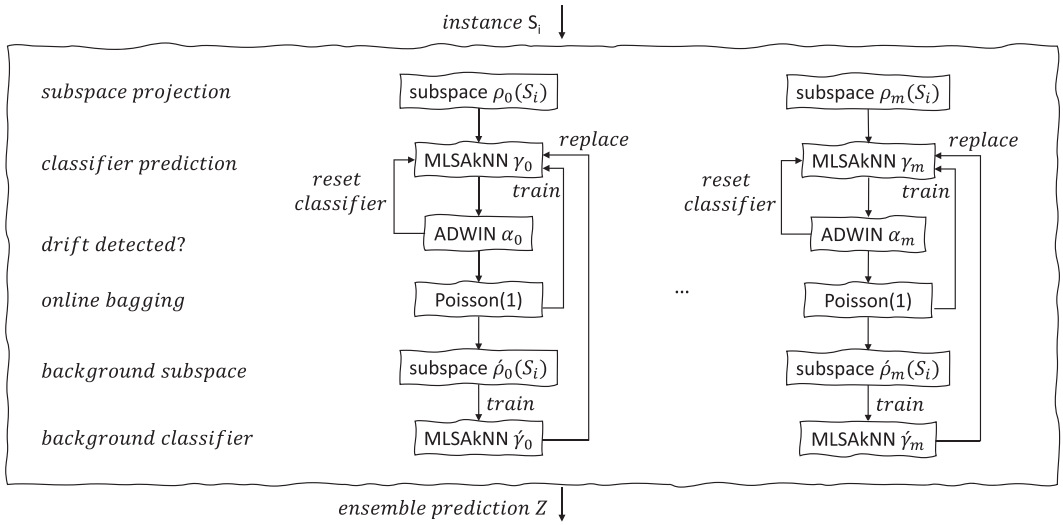


Fig. 1. Flowchart of Adaptive Ensemble of Self-Adjusting Nearest Neighbor Subspaces.

As shown before, there are many advantages of ensemble learning for both the data streaming and multi-label problem domains. AESAKNNS leverages the Multi-Label Self-Adapting kNN (MLSAkNN) [11] classifier that is individually capable of adjusting for concept drift and other changes in multi-label data. MLSAkNN self-adjusts the k neighbors value over time. However, it is well known the sensitivity of the k neighbors and the feature space organization on the distance calculation. A change in the feature subspace composition has a strong influence in the selection of the k neighbors and thus on the predictions. Similarly, the different instance subsets presented to each of the base classifiers also influence the predictions. This feature and instance space heterogeneity motivates our proposal to employ adaptive random subspaces and online bagging on the training of the base classifiers to increase their diversity. This allows for the ensemble to avoid the traditional pitfalls of problem transformation methods. Moreover, since each MLSAkNN base learner is trained on different subsets of features and instances, each classifier contains potentially novel and diverse information on the problem. Contrary to other random subspaces approaches, such as Adaptive Random Forests [5] which employs a fixed-size subset of features (all base classifiers have the same constant number of input features and they are invariant), we propose varying-size adaptive random subspaces (all base classifiers have a different number of input features and the subsets of features change over time). AESAKNNS employs a number of methodologies to combat concept drift, including a collection of ADWIN detectors that watch each base classifier. This mechanism serves to warn the ensemble that a concept drift was detected on the particular feature subspace. On detection of concept drift the ensemble immediately resets the learning on active classifiers, to potentially capture the new concept with existing feature spaces. However, after drift detection, AESAKNNS will simultaneously initialize a set of background classifiers on new subsets of features. The background ensemble is not included in the prediction but will train in tandem with the active ensemble over a duration of instances defined as the window size w. Once the background ensemble has trained on w instances, AESAKNNS analyzes the Hamming score and subset accuracy of all classifiers from both ensembles. The best performing base classifiers are then selected to occupy the new primary ensemble and the background classifier is cleared. This way, the most relevant features and concepts are constantly being evaluated and replaced to reflect the current data in the stream as in dynamic classifier selection.

3.1. MLSAkNN as a base classifier

MLSAkNN is a multi-label algorithm designed by Roseberry et al. [11]. It is an adaption of traditional kNN for the multi-label streaming environment. In order to adapt to concept drift, the algorithm includes a variable size window, whose size expands and contracts based on the current drift detection. This allows the algorithm to dynamically choose which parts of the instance stream are important and remove old instances in case of concept drift. MLSAkNN also utilizes a punitive system to attempt removing instances that contribute heavily to errors. Due to the nature of multi-label data, the punitive system can choose to penalize instances for each label individually. This allows for data adaption that is tailored to each label stream and beneficial for overall performance. Finally, MLSAkNN employs an adaptive k value for selecting the quantity of nearest neighbors. The adaptive k exists for each label, providing a more autonomous algorithm without the need for manual parameter tuning. MLSAkNN is a very suitable base classifier for our ensemble approach where instances presented to the classifiers are subject to adaptive feature subspace projections and online bagging. This way, we increase the diversity of the classifiers both in the feature and instance space, leading to better predictions than individual classifiers on homogeneous instance and feature spaces.

3.2. Online bagging and instance subsets

Bagging is the process of training a set of weak learners in parallel to unique collections of instance subsets with oversampling. In our case, we utilize an online bagging following a $Poisson(\lambda)$ distribution with $\lambda = 1$ in order to distribute examples to different weak learners in the ensemble. $Poisson(1)$ represents the converging binomial distribution of examples as the number of instances grows very large. This method applies additional weight to instances for specific learners, allowing us to sample with replacement from our data stream. Online bagging has been successfully used in Leveraging Bagging [45], Adaptive Random Forest [5], and Kappa Updated Ensemble [8]. Online bagging increases the diversity of the instance set presented to the base classifiers, and in the case of MLSAkNN it directly affects the neighborhood selection of the k closest instances. Online bagging is applied in lines 21 and 38 of Alg. 1 to update the active and background ensembles

base classifiers respectively. Similarly, it can be observed in the flowchart in Fig. 1 before the update of the base classifiers.

3.3. Adaptive random feature subspaces

An important aspect of supervised learning is the ability to identify relevant features that impact predictions. In the multi-label context, this is true for each individual label and groups of labels, but not all of the attributes are relevant for all of the labels. Therefore, it is a good idea to specialize the base classifiers on heterogeneous feature subspaces. We leverage the diversity of each classifier to solve this issue. Each base classifier within the ensemble is given a unique subset of features to consider. Traditional approaches such as Adaptive Random Forest [5] employ a fixed-size subspace that remains invariant for the whole of the streams. This constrains the algorithm's ability to adapt to better feature subspaces over time. Our contribution here is to dynamically change the size and composition of the feature subspace over time. This way, we allow each of the base classifiers in the ensemble to adapt to bigger or smaller feature subspaces. Every time we learn a new ensemble in the background the base classifiers will explore a new random subspace of features of varying sizes. The subspace size is modeled using a normal distribution with a mean 70% of the number of features. Therefore, during evaluations of the ensemble, classifiers with poor performance are discarded in favor of others with better performance in other feature subspaces. Because new classifiers are constantly being created and evaluated throughout the duration of the data stream, AESAKNNS has the means to adapt to changes in relevant features over time effectively. This adds to the overall diversity of the ensemble and helps to boost overall performance. Lines 7 and 31 of Alg. 1 show the definition of the random feature subspaces.

3.4. Concept drift detection using ADWIN

ADWIN [46] is an adaptive windowing algorithm that detects changes in data distributions over a certain number of examples. Changes are determined by examining the statistical properties for two sub portions of a given window and determining if there is a significant difference in mean measurements. Many algorithms and detection methods in the literature follow this implementation strategy [5,46,45]. In AESAKNNS, ADWIN is used as the primary source of explicit drift detection on each of the base classifiers that operate on different subsets of features of the stream. This way, ADWIN will warn about potential changes of the data distribution that may not necessarily affect all of the stream but only to subsets of features and labels. Upon detection of a drift the base classifier on that feature subset is reset and it triggers the learning a new background ensemble. ADWIN is employed in lines 12–19 of Alg. 1.

3.5. Background ensemble to adapt to concept drift

Upon drift detection using ADWIN on any of the classifiers, a new ensemble is initialized in the background. This new ensemble is trained in parallel with the currently active one. The base classifier on the feature space that triggered the drift is reset. However, our contribution here is that the new background ensemble will learn new base classifiers on new varying-size feature subspaces, potentially detecting new more relevant features after the drift. The differences in feature space and instance distribution will lead to different knowledge products after a predefined duration. After this period, a comparison is performed between the active and background ensemble where only the best performing classifiers are selected for the new active ensemble. The criterion to select the best performing classifiers is a linear combination of the subset accuracy and the Hamming score. These two metrics aim at different objectives in the multi-label classification, starting by ensuring that new concepts are being detected and added to the ensemble, maintaining still relevant knowledge, and removing outdated base classifiers with old concepts. This replacement strategy to keep the fittest base classifiers while removing weakest ones helps to quickly adapt to concept drift. Lines 28–34 of Alg. 1 show the background ensemble initialization, lines 35–43 show the background ensemble update, and lines 44–47 show the ensemble replacement.

4. Experimental study

This section presents the experimental study and comparison with works in the state of the art. The experiments are designed to answer the following research questions:

- **RQ1:** Can AESAKNNS demonstrate competitive performance compared to state of the art classification methods for multi-label data streams?
- **RQ2:** Is AESAKNNS a competitive ensemble when compared strictly to other ensemble algorithms?
- **RQ3:** Is AESAKNNS competitive against other kNN adaptations for multi-label data streams?
- **RQ4:** How does each of the AESAKNNS contributions help to improve the classification performance?

4.1. Experimental setup

4.1.1. Algorithms

Table 1 presents a taxonomy of the multi-label algorithms used in our experimental study, including Binary Relevance (BR) and Algorithm Adaptation (AA) methods. All algorithms are implemented in Java and publicly available in MOA [47]. The source code

Table 1
Taxonomy of algorithms used in the experiments.

Family	Ref	Acronym	Algorithm
BR + Single	[48]	NB	Naive Bayes
BR + Single	[49]	HT	Hoeffding Tree
BR + Single	[50]	AHT	Adapting Hoeffding Option Tree
BR + Single	[51]	SCD	Single Classifier Drift
BR + Ensemble	[45]	LB	Leveraging Bag
BR + Ensemble	[15]	OB	Oza Bag
BR + Ensemble	[48]	OBA	Oza Bag Adwin
BR + Ensemble	[15]	OBO	Oza Boost
BR + Ensemble	[48]	OBOA	Oza Boost Adwin
BR + Ensemble	[52]	OCB	Online Coordinated Boosting
BR + Ensemble	[53]	DWM	Dynamic Weighted Majority
BR + Ensemble	[54]	AUE	Accuracy Updated Ensemble
BR + Ensemble	[5]	ARF	Adaptive Random Forest
BR + kNN	[55]	kNN	kNN
BR + kNN	[56]	kNNP	kNN PAW
BR + kNN	[56]	kNNPA	kNN PAW ADWIN
BR + kNN	[57]	SAMkNN	Self-Adjusting Memory kNN
AA + Incremental	[58]	BRU	Binary Relevance Updateable
AA + Incremental	[58]	CCU	Classifier Chains Updateable
AA + Incremental	[58]	PSU	Pruned Sets Updateable
AA + Incremental	[58]	RTU	Ranking Threshold Updateable
AA + Incremental	[59]	MLHT	Multilabel Hoeffding Tree
AA + Incremental	[60]	AMR	Adaptive Model Rules
AA + Ensemble	[58]	BML	Bagging ML Updateable
AA + Ensemble	[61]	OBML	Oza Bag ML
AA + Ensemble	[61]	OBAML	Oza Bag Adwin ML
AA + kNN	[14]	MLkNN	ML kNN
AA + kNN	[21]	MLSAMkNN	ML Self-Adjusting Memory kNN
AA + kNN	[22]	MLSAMPkNN	ML Self-Adjusting Memory Punitive kNN
AA + kNN	[11]	MLSAkNN	ML Self-Adjusting kNN
AA + Ensemble + kNN	–	AESAKNNS	Adaptive Ensemble of SAkNN Subspaces

Table 2
Datasets and their characteristics.

Dataset	Instances	Features	Labels	Cardinality	Density
Birds	645	260	19	1.01	0.05
Virus	207	749	6	1.22	0.20
Flags	194	19	7	3.39	0.48
Scene	2,407	294	6	1.07	0.18
Enron	1,702	1,001	53	4.27	0.08
Genbase	662	1,186	27	1.25	0.05
Medical	978	1,449	45	1.25	0.03
Water-qual	1,060	16	14	5.07	0.36
Corel-5 k	5,000	499	374	3.52	0.01
Eukaryote	7,766	440	22	1.15	0.05
Plant	978	440	12	1.08	0.09
Reuters	6,000	500	103	0.11	0.01
Mediamill	43,907	120	101	4.38	0.04
Ohsumed	13,929	1,002	23	0.81	0.04
CAL-500	502	68	174	26.04	0.15
Yelp	10,806	671	5	1.64	0.33
Slashdot	3,782	1,079	22	1.18	0.05
Human	3,106	440	14	1.19	0.08
Langlog	1,460	1,004	75	15.94	0.21
Gnegative	1,392	440	8	1.05	0.13
CHD	555	49	6	2.58	0.43
Stackex	1,675	585	227	2.41	0.01
Corel-16 k	13,766	500	153	2.86	0.02
Imdb	120,919	1,001	28	1.00	0.04
Nuswide-C	269,648	129	81	1.87	0.02
Nuswide-B	269,648	501	81	1.87	0.02
Yahoo-soc	14,512	31,802	27	1.67	0.06
Eurlex	19,348	5,000	201	2.21	0.01
Hypersphere	100,000	100	10	2.31	0.23
Hypercube	100,000	100	10	1.00	0.10

of our AESAKNNS is publicly available at https://github.com/canoalberto/AESAKNNS to facilitate the reproducibility of the research. The table presents all of the models that were run during experimentation, along with the families they belong to. Within the exhaustive list of 30 classifiers, particular importance are the ensemble and kNN families of algorithms. Ensemble methods have several options for enhancing performance and combating concept drift, and all are run using 10 base classifiers. No individual hyperparameter optimization was conducted for any algorithm as we believe algorithms should exhibit a robust performance off the shelf. All algorithms were compared across the same 12 multi-label prequential metrics, which will be discussed in Section 4.1.3.

4.1.2. Datasets

Multi-label datasets used in our experiments cover a wide range of properties. We evaluate the performance of the algorithms on 30 datasets of up to 269.648 k instances, 31.8 k features, and 374 labels. The data properties of each individual dataset are shown in Table 2. These measures include the number of instances, features, and labels. Other measures include cardinality, which measures the average amount of labels per instance, and density, calculated as cardinality divided by the number of labels. The lower the density the more sparse positive labels are through the dataset.

4.1.3. Metrics

Due to the incremental nature of data streams, traditional methodologies for evaluation such as cross-validation are unusable. This gives rise to the use of prequential evaluations to measure model performance [62]. When calculating subset accuracy and Hamming score for AESAKNNS, we must account for the total and partial correctness of the multi-label prediction. In order to do so, we utilize the following definitions over n instances and L labels [2].

Example-based metrics evaluate the difference between the actual and predicted labelsets, averaged over n instances. For a true labelset $Y_i = \{y_{i1} \dots y_{iL}\}$ and a predicted labelset $Z_i = \{z_{i1} \dots z_{iL}\}$, the example-based metrics are:

$$Subset\ Accuracy = \frac{1}{n}\sum_{i=0}^{n} \mathbb{1} \mid \mathbb{Y}_i = \mathbb{Z}_i$$

$$Hamming\ Score = \frac{1}{nL}\sum_{i=0}^{n}\sum_{l=0}^{L} \mathbb{1} \mid \mathbb{y}_{il} = \mathbb{z}_{il}$$

$$Example - based\ Accuracy = \frac{1}{n}\sum_{i=0}^{n} \frac{|\mathbb{Y}_i \cap \mathbb{Z}_i|}{|\mathbb{Y}_i \cup \mathbb{Z}_i|}$$

$$Example - based\ Precision = \frac{1}{n}\sum_{i=0}^{n} \frac{|\mathbb{Y}_i \cap \mathbb{Z}_i|}{|\mathbb{Z}_i|}$$

$$Example - based\ Recall = \frac{1}{n}\sum_{i=0}^{n} \frac{|\mathbb{Y}_i \cap \mathbb{Z}_i|}{|\mathbb{Y}_i|}$$

$$Example - based\ F1 = \frac{1}{n}\sum_{i=0}^{n} \frac{2\times|\mathbb{Y}_i \cap \mathbb{Z}_i|}{|\mathbb{Y}_i| + |\mathbb{Z}_i|}$$

Label-based metrics are averaged over all labels and are best defined using the number of true positives, true negatives and false negatives for each label l. Formally, true positives $tp_l = \sum_{i=0}^{n} \mathbb{1} | \mathbb{y}_{il} = \mathbb{z}_{il} = \mathbb{1}$, false positives $tn_l = \sum_{i=0}^{n} \mathbb{1} | \mathbb{y}_{il} = \mathbb{0}, \mathbb{z}_{il} = \mathbb{1}$, and false negatives $fn_l = \sum_{i=0}^{n} \mathbb{1} | \mathbb{y}_{il} = \mathbb{1}, \mathbb{z}_{il} = \mathbb{0}$. Macro-averaged metrics are calculated and then averaged over all labels, as:

$$Macro - averaged\ Precision = \frac{1}{L}\sum_{l=0}^{L} \frac{tp_l}{tp_l + fp_l}$$

$$Macro - averaged\ Recall = \frac{1}{L}\sum_{l=0}^{L} \frac{tp_l}{tp_l + fn_l}$$

Micro-averaged metrics are averaged over all labels and instances, formally:

$$Micro - averaged\ Precision = \frac{\sum_{l=0}^{L} tp_l}{\sum_{l=0}^{L} tp_l + \sum_{l=0}^{L} fp_l}$$

$$Micro - averaged\ Recall = \frac{\sum_{l=0}^{L} tp_l}{\sum_{l=0}^{L} tp_l + \sum_{l=0}^{L} fn_l}$$

As precision can often be optimized at the expense of recall, and vice versa, the F1 is preferred to balance the two. Like the example-based F1, the micro- and macro-averaged F1 are computed by taking the harmonic mean of the precision and recall. Subset accuracy represents the exact match for all of the labels in the labelset, a very strict metric and preferred to best compare algorithms. Hamming score calculates the successful predictions per instance and label, i.e., the symmetric difference, however, it suffers from high imbalance.

4.2. Overall comparison with all classifiers on all metrics

The first experiment evaluates and compares the overall performance of the 30 algorithms on the 30 datasets for all of the 12 metrics to address RQ1. Table 3 collects the results in a compact table, showing the average metric values for all the datasets, along with the rank of the algorithm according to Friedman (the lower the rank, the better). Based on this, one can observe that AESAKNNS was the top performing model for 9 of the 12 metrics, with significant differences between AESAKNNS's performance and the sec-

Table 3
Performance of 12 metrics for all algorithms across all 30 datasets and ranks.

Algorithm	Su. Acc	H. Sco	Ex. Acc	Ex. Pre	Ex. Rec	Ex. F1	Mi. Pre	Mi. Rec	Mi. F1	Ma. Pre	Ma. Rec	Ma. F1	Rank
NB	0.139	0.833	0.230	0.311	0.421	0.312	0.389	0.419	0.303	0.198	0.302	0.198	21.17
HT	0.209	0.914	0.292	0.525	0.330	0.374	0.596	0.319	0.370	0.230	0.179	0.179	19.13
AHT	0.220	0.916	0.302	0.538	0.339	0.384	0.607	0.328	0.379	0.240	0.182	0.186	15.96
SCD	0.212	0.875	0.304	0.416	0.442	0.380	0.490	0.435	0.374	0.278	0.284	0.247	14.75
LB	0.272	0.815	0.377	0.530	0.552	0.473	0.489	0.540	0.433	0.262	0.366	0.248	9.58
OB	0.202	0.827	0.289	0.499	0.415	0.383	0.465	0.408	0.351	0.219	0.281	0.194	18.83
OBA	0.224	0.832	0.312	0.520	0.438	0.406	0.480	0.427	0.373	0.245	0.291	0.213	14.58
OBO	0.120	0.728	0.250	0.339	0.553	0.344	0.318	0.547	0.328	0.251	0.416	0.269	16.58
OBOA	0.026	0.649	0.176	0.211	**0.596**	0.271	0.202	0.588	0.260	0.197	0.439	0.240	20.75
OCB	0.309	0.929	0.415	0.613	0.460	0.505	0.634	0.445	0.494	0.336	0.265	0.282	5.50
DWM	0.196	0.886	0.291	0.428	0.443	0.373	0.488	0.434	0.361	0.231	0.275	0.215	16.96
AUE	0.052	0.543	0.158	0.270	0.565	0.244	0.225	**0.571**	0.209	0.139	**0.510**	0.166	22.50
ARF	0.298	0.863	0.407	0.576	0.544	0.507	0.553	0.529	0.477	0.326	0.342	0.283	7.17
kNN	0.238	0.912	0.322	0.496	0.365	0.395	0.545	0.353	0.391	0.237	0.182	0.186	15.38
kNNP	0.240	0.914	0.321	0.505	0.361	0.394	0.558	0.348	0.390	0.249	0.178	0.187	14.83
kNNPA	0.259	0.918	0.348	0.529	0.392	0.424	0.582	0.377	0.418	0.268	0.193	0.203	11.33
SAMkNN	0.263	0.923	0.343	0.594	0.376	0.424	0.664	0.359	0.415	0.313	0.183	0.207	10.25
BRU	0.212	0.913	0.300	0.493	0.343	0.383	0.515	0.331	0.373	0.198	0.184	0.174	19.33
CCU	0.227	0.914	0.308	0.506	0.346	0.391	0.538	0.332	0.382	0.204	0.185	0.177	16.71
PSU	0.161	0.879	0.233	0.300	0.274	0.279	0.310	0.255	0.270	0.102	0.134	0.109	26.63
RTU	0.165	0.892	0.156	0.279	0.157	0.174	0.467	0.151	0.171	0.156	0.098	0.100	27.75
MLHT	0.160	0.879	0.232	0.299	0.273	0.278	0.308	0.254	0.269	0.102	0.134	0.109	27.21
AMR	0.139	0.833	0.230	0.312	0.421	0.312	0.391	0.419	0.303	0.197	0.301	0.198	21.08
BML	0.221	0.915	0.306	0.537	0.347	0.389	0.565	0.334	0.379	0.243	0.180	0.183	15.67
OBML	0.160	0.880	0.231	0.306	0.271	0.280	0.314	0.251	0.268	0.095	0.127	0.102	27.42
OBAML	0.223	0.893	0.286	0.393	0.324	0.339	0.405	0.304	0.326	0.148	0.153	0.136	23.25
MLkNN	0.230	0.914	0.304	0.547	0.337	0.378	0.605	0.324	0.375	0.248	0.167	0.181	16.50
MLSAMkNN	0.316	0.925	0.404	0.587	0.443	0.482	0.620	0.427	0.475	0.318	0.225	0.245	7.46
MLSAMPkNN	0.338	0.924	0.435	0.573	0.480	0.514	0.587	0.461	0.504	0.316	0.248	0.267	6.33
MLSAkNN	0.352	0.937	0.466	0.622	0.522	0.552	0.628	0.504	0.541	0.367	0.313	0.326	3.42
AESAKNNS	**0.389**	**0.940**	**0.504**	**0.683**	0.552	**0.593**	**0.702**	0.536	**0.583**	**0.429**	0.323	**0.351**	**2.00**

ond best algorithm in many of the metrics. For example, subset accuracy is 3 points better (10% improvement) for AESAKNNS compared to the second best MLSAkNN. This shows that the ensemble strategy and the original contributions with the feature subspaces, instance subsets, and background ensembles lead to a significant improvement compared to the single base classifier. While AESAKNNS's recall values may not be the best, the F1 metric, which balances precision and recall, clearly favors AESAKNNS as a well rounded classifier. The following best-performing ensemble classifiers are Online Coordinate boosting (OCB) and Adaptive Random Forest (ARF). Complete results for all classifiers on all individual datasets on all metrics are available as supplementary material on the Github repository at https://github.com/canoalberto/AESAKNNS.

Fig. 2 show the Bonferroni-Dunn statistical test for all algorithms on the subset accuracy as the most representative metric. The figure illustrates the rank of the algorithms according to the subset accuracy metric and the critical distance for significance $\alpha = 0.01$. Algorithms outside the interval of the critical distance are said to perform statistically worse than the control method.

4.3. Ensembles comparison

The second experiment evaluates specifically the performance of ensemble classifiers to address RQ2. Table 4 presents subset accuracy measures for all ensemble algorithms against all datasets. Complete results are available on the Github at https://github.com/canoalberto/AESAKNNS for all datasets and metrics. AESAKNNS outperforms all of the ensemble methods and obtains the best subset accuracy for 18 of the 30 datasets, providing the highest average accuracy and the best rank. The second best ensemble on average subset accuracy is Online Coordinate boosting (OCB), but there is an 8-points difference with our proposed method, whereas the second best according to the rank is Adaptive Random Forest (ARF). On the other hand, the worst performing ensembles are Oza Boost Adwin and Accuracy Updated Ensemble.

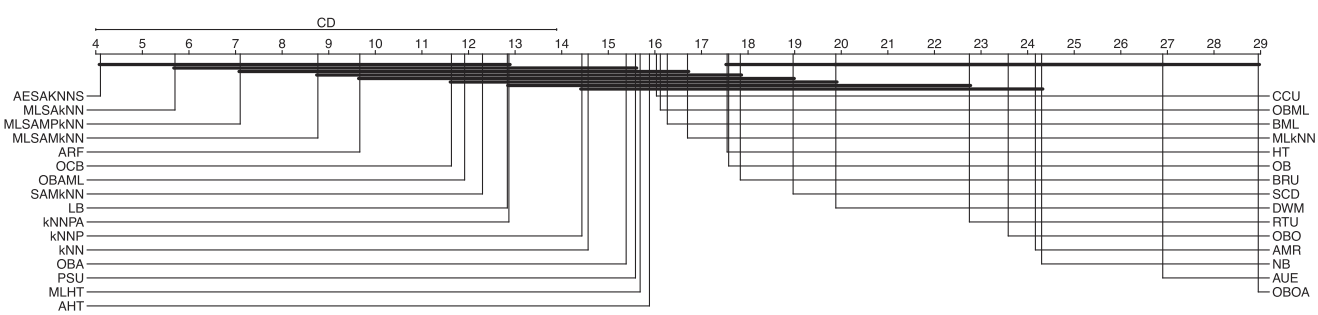


Fig. 2. Bonferroni-Dunn for subset accuracy on all classifiers.

Table 4
Subset accuracy for all ensemble classifiers on each dataset.

Dataset	LB	OB	OBA	OBO	OBOA	OCB	DWM	AUE	ARF	BML	OBML	OBAML	AESAKNNS
Birds	0.418	0.440	0.440	0.192	0.185	0.421	0.364	0.071	0.458	**0.425**	0.451	0.451	0.454
Virus	0.304	0.146	0.146	0.254	0.128	0.395	0.217	0.000	0.315	0.350	0.259	0.259	**0.671**
Flags	0.103	0.115	0.115	0.080	0.080	0.114	0.104	0.009	0.118	0.105	**0.129**	**0.129**	0.123
Scene	0.662	0.349	0.515	0.517	0.045	0.837	0.497	0.062	0.795	0.357	0.343	0.493	**0.875**
Enron	0.032	0.036	0.031	0.003	0.001	0.067	0.034	0.001	0.055	0.083	0.088	**0.109**	0.095
Genbase	0.744	0.365	0.365	0.016	0.016	0.765	0.234	0.121	0.452	0.370	0.262	0.238	**0.918**
Medical	0.475	0.276	0.276	0.000	0.000	0.262	0.001	0.197	**0.503**	0.285	0.148	0.148	0.353
Water-qual	0.017	0.005	0.006	0.002	0.002	0.010	0.002	0.001	0.019	0.004	0.007	0.012	**0.027**
Corel-5 k	0.021	0.007	0.008	0.000	0.000	0.032	0.000	0.000	0.040	0.009	0.009	0.023	**0.055**
Eukaryote	0.626	0.406	0.505	0.345	0.000	0.703	0.431	0.195	0.686	0.406	0.162	0.752	**0.813**
Plant	0.317	0.229	0.248	0.119	0.000	0.726	0.238	0.028	0.430	0.236	0.314	0.539	**0.762**
Reuters	0.080	0.065	0.065	0.000	0.000	0.052	0.012	0.036	0.147	0.081	0.211	0.191	**0.240**
Mediamill	0.117	0.069	0.063	0.029	0.000	0.095	0.003	0.055	0.146	0.071	0.053	0.083	**0.190**
Ohsumed	0.138	0.152	0.152	0.123	0.016	0.132	0.127	0.042	0.179	0.124	0.162	0.157	**0.680**
CAL-500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Yelp	0.533	0.340	0.459	0.461	0.022	0.446	0.402	0.166	0.631	0.347	0.238	0.413	**0.665**
Slashdot	0.026	0.102	0.102	0.000	0.000	0.061	0.029	0.058	0.096	0.011	0.130	0.137	**0.143**
Human	0.510	0.197	0.300	0.376	0.038	0.636	0.364	0.054	0.536	0.198	0.207	0.648	**0.751**
Langlog	0.025	0.027	0.027	0.000	0.000	0.146	0.154	0.007	0.146	0.139	0.138	0.139	**0.178**
Gnegative	0.630	0.581	0.597	0.242	0.000	0.887	0.605	0.180	0.735	0.602	0.416	0.416	**0.896**
CHD	0.184	0.156	0.156	0.129	0.124	0.145	0.106	0.006	0.171	0.155	0.143	0.143	**0.229**
Stackex	0.007	0.016	0.016	0.000	0.000	0.020	0.012	0.006	0.004	0.002	**0.028**	0.011	0.014
Corel-16 k	0.058	0.013	0.014	0.034	0.000	0.087	0.001	0.001	0.101	0.022	0.021	0.075	**0.122**
Imdb	0.012	0.003	0.007	0.013	0.003	0.030	0.012	0.000	0.029	0.021	0.108	**0.110**	0.093
Nuswide-C	0.263	0.240	0.250	0.198	0.100	0.213	0.176	0.221	**0.269**	0.240	0.133	0.249	0.240
Nuswide-B	0.226	0.226	0.225	0.197	0.013	0.211	0.227	0.004	**0.254**	0.220	0.118	0.248	0.239
Yahoo-soc	0.004	0.020	0.006	0.000	0.000	0.102	0.027	0.002	0.002	0.144	**0.282**	**0.282**	0.186
Eurlex-sm	0.000	0.000	0.000	0.000	0.000	0.073	0.001	0.000	0.000	**0.158**	0.141	0.135	0.118
Hypersphe	0.622	0.472	**0.628**	0.182	0.000	0.611	0.493	0.014	0.622	0.464	0.029	0.029	0.529
Hypercube	0.997	0.994	0.998	0.080	0.000	0.999	0.995	0.030	0.998	0.993	0.068	0.068	**0.999**
Average	0.272	0.202	0.224	0.120	0.026	0.309	0.196	0.052	0.298	0.221	0.160	0.223	**0.389**
Rank	5.68	7.68	6.55	9.98	12.23	4.77	8.27	11.19	3.84	6.89	6.85	5.00	**2.06**

Figs. 3–5, show the Bonferroni-Dunn test for ensemble algorithms on the subset accuracy, Hamming score, and example-based F1 respectively. The figures illustrate the rank of the algorithms and the critical distance for $\alpha = 0.01$. According to the test, it cannot be said that there are statistically significant differences when comparing AESAKNNS with Adaptive Random Forest (ARF)

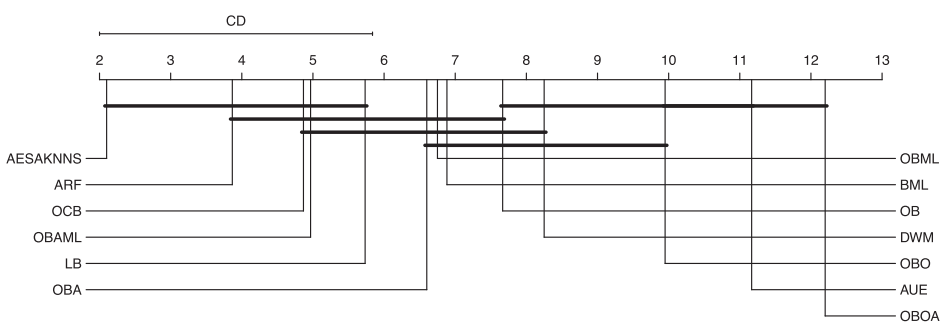


Fig. 3. Bonferroni-Dunn for subset accuracy on ensembles.

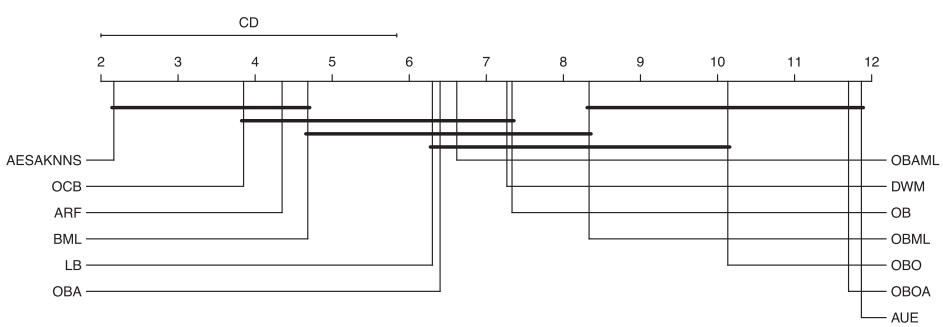


Fig. 4. Bonferroni-Dunn for Hamming score on ensembles.

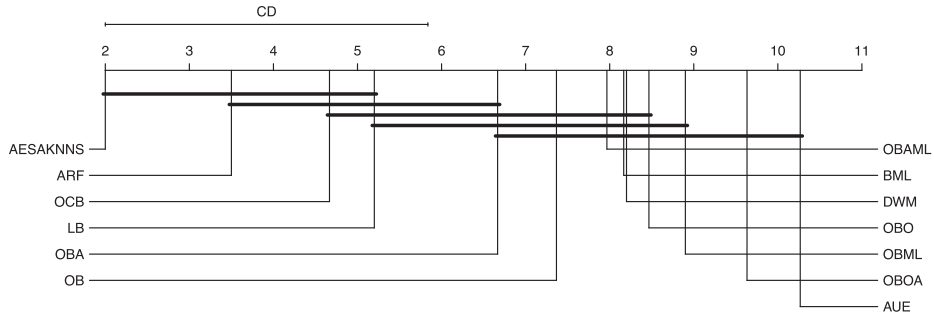


Fig. 5. Bonferroni-Dunn for Hamming score on example-based F1.

and Online Coordinated Boosting (OCB) for the three metrics. However, statistically significant differences exist for with rest of the ensembles.

Table 5 shows the *p*-values reported for the Wilcoxon signed-rank test on the main performance metrics: subset accuracy, Hamming score, example-based F1, micro-F1, and macro-F1, where *p*-values < 0.01 indicate statistically significant differences between AESAKNNS and the compared method (pairwise comparison test). The smaller the *p*-value, the higher confidence. There are statistically significant differences concerning all the ensembles for all the metrics, except for the macro-F1 with the Adaptive Random Forest classifer.

Fig. 6 illustrates the Bayesian sign test on the subset accuracy and example-based F1 as the most representative metrics. This test returns probabilities that one model will outperform the other based on measured performance. The top region indicates practical equivalence, while the lower right portion denotes better performance for AESAKNNS and the remaining side for the opposing algorithm. We can see that AESAKNNS outperforms Adaptive Random Forest at almost every evaluation and Online Coordinated Boosting every time.

Figs. 7–10, help to visualize why AESAKNNS is rating well across these various tests and metrics. Here we see how subset accuracy and example-based F1 vary with the processing of the online instances over time (prequential metrics), adapting to concept drift and changing properties of the stream. Notice how AESAKNNS is the first to adapt and maintains the highest subset accuracy and F1 for the most prolonged duration. This can contribute to an explanation as to why we see such performance gains with our method. First, we adapt early. As soon as drifts are detected, changes are made, and metrics for meta evaluations begin to be gathered. Second, our ensemble strongly promotes diversity of instance and feature spaces, and contains competent classifiers for longer amounts of time due to frequent evaluations and updates. In short, our method is more adaptable and recovers faster to changes in data concepts.

Table 5
Wilcoxon signed test: AESAKNNS vs ensembles (*p*-values).

Algorithm	Su. Acc	H. Sco	Ex. F1	Mi. F1	Ma. F1
LB	3.92E-05	4.46E-05	2.22E-05	8.20E-06	2.64E-04
OB	1.24E-06	7.47E-06	3.49E-06	1.06E-06	1.42E-05
OBA	7.82E-06	2.04E-05	5.12E-06	1.44E-06	3.16E-05
OBO	9.13E-07	8.67E-07	8.67E-07	8.67E-07	6.91E-04
OBOA	9.13E-07	8.67E-07	8.67E-07	8.67E-07	5.57E-04
OCB	7.13E-06	1.11E-04	3.85E-06	3.85E-06	2.45E-04
DWM	9.13E-07	9.60E-07	1.18E-06	1.30E-06	9.86E-06
AUE	9.13E-07	8.67E-07	1.30E-06	8.67E-07	4.09E-05
ARF	2.36E-04	3.32E-04	5.18E-04	4.46E-05	1.22E-02
BML	2.74E-06	3.30E-05	1.59E-06	1.06E-06	1.94E-06
OBML	3.60E-05	2.61E-06	2.36E-06	1.94E-06	8.67E-07
OBAML	2.96E-04	4.23E-06	1.76E-06	1.76E-06	8.67E-07

4.4. Nearest neighbors comparison

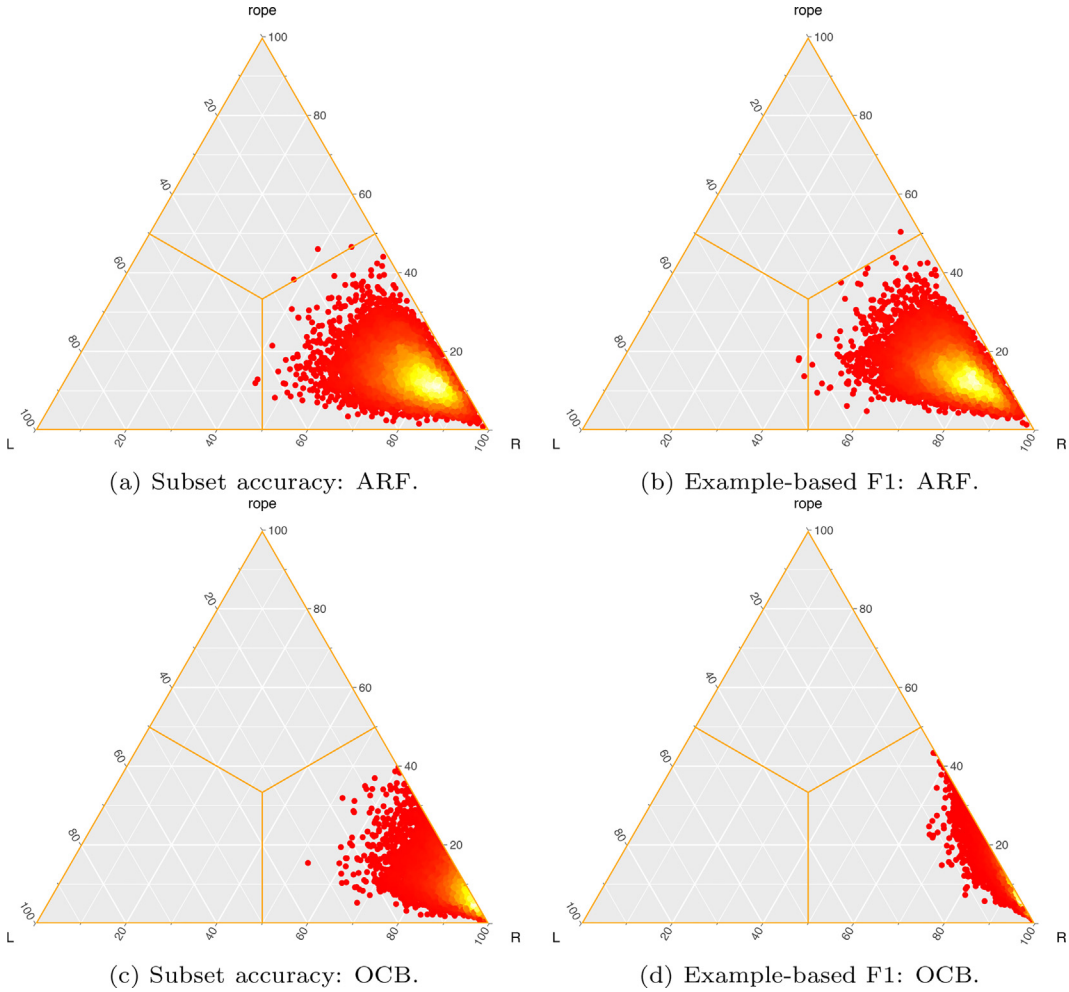
The third experimental study aims at providing an in-depth evaluation and comparison of classifiers based on nearest neighbors to address RQ3. Table 6 presents the subset accuracy for the nearest neighbors methods on all datasets. Complete results are available for all datasets and metrics on the Github at https://github.com/canoalberto/AESAKNNS. We observe that AESAKNNS outperforms the kNN family of algorithms on 17 out of the 30 datasets. It is also shown to have the most competitive average accuracy and the best rank. Showing that at a minimum AESAKNNS is a top running contender not only for ensemble algorithms but also adapted kNN algorithms. AESAKNNS shows significantly better performance than MLSAkNN, demonstrating the advantages of the ensemble approach and its methodologies for adaptation to concept drift using adaptive feature subspaces and online bagging.

Figs. 11–13, show the Bonferroni-Dunn for subset accuracy, Hamming score, and example-based F1. The test indicates it cannot find significant statistical differences exist between AESAKNNS and MLSAkNN, MLSAMPkNN, and MLSAMkNN based on subset accuracy and F1, which means that a significant statistical difference exists for the remaining kNN methods. The test shows the significant advantage of the ensemble approach compared to single classifiers.

Table 7 presents the Wilcoxon test between AESAKNNS and the nearest neighbors methods. It reveals something more interesting in the kNN family compared to the ensemble test. While most values in the table confirm with confidence that there are statistically significant differences, we see that MLSAkNN maintains relatively low *p*-values except for the macro-averaged F1. As the base classifier of AESAKNNS, it is understandable that these values would be the most similar. It serves as an important reflection to note that the *p*-values for MLSAkNN should help establish the statistical difference of AESAKNNS based on only the ensemble mechanisms.

Fig. 14 shows the Bayesian analysis for subset accuracy and example-based F1, and compares probabilities of outperformance between AESAKNNS and MLSAkNN as well as AESAKNNS and MLSAMPkNN. The big takeaway from this test is to reveal that AESAKNNS is either outperforming or equivalent to the opposing algorithms. However, it is notable that by simply existing in the same family of algorithms it is easier to achieve statistical indifference. This is even more noticeable for the MLSAkNN, also acting as the base classifier for AESAKNNS. These figures also reveal more support for the opposing algorithms than the ARF and OCB models, however, the volume of supporting points for this is minimal. Therefore, this test supports the better performance of the our diverse ensemble approach.

Figs. 15–18 compare nearest neighbor approaches showing the prequential subset accuracy and example-based F1 on four datasets with the increasing number of processed instances. AESAKNNS



(a) Subset accuracy: ARF. (b) Example-based F1: ARF.

(c) Subset accuracy: OCB. (d) Example-based F1: OCB.

Fig. 6. Bayesian sign test on top ensembles (AESAKNNS vs ARF and OCB).

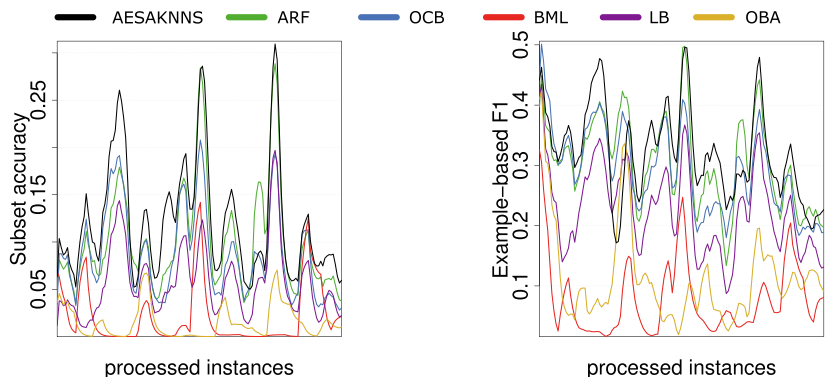


Fig. 7. Ensembles comparison: prequential subset accuracy and F1 on Corel-16 k.

maintains the best performance over the most prolonged interval of instances. It is interesting to highlight the performance in Figs. 15,16, where AEKANNS does not experience a significant drop in the metrics while kNNPA, MLkNN, and kNN do. Moreover, in Fig. 18 we observe four main concept drifts which result in four significant drops in the metrics. However, AEKANNS experiences a smaller drop and is able to maintain the highest accuracy and F1 during the drift, while quickly recovering and adapting to the new concept.

4.5. Contributions of AESAKNNS

The fourth experiment performs an ablation study and aims at evaluating how each of the main four contributions of AESAKNNS help to improve the classification and by what margin. Figs. 19–22, show the prequential subset accuracy and example-based F1 over time for four example datasets. It illustrates the performance of AESAKNNS and four variants of the algorithm without each of these contributions: no background ensemble, no adaptive feature

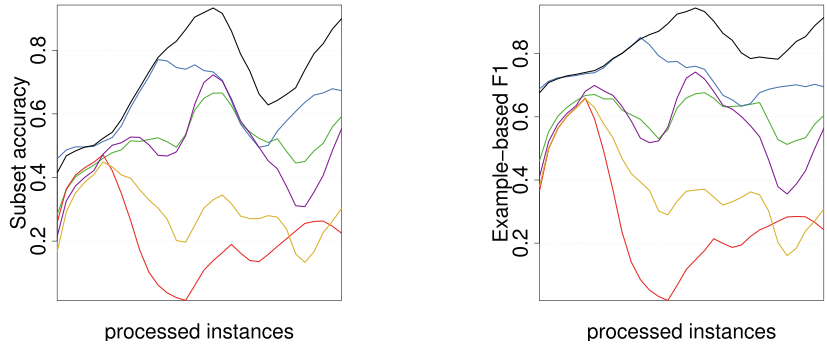


Fig. 8. Ensembles comparison: prequential subset accuracy and F1 on Human.

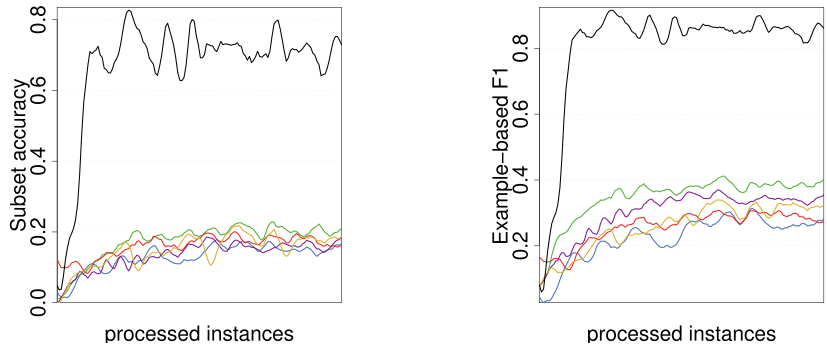


Fig. 9. Ensembles comparison: prequential subset accuracy and F1 on Ohsumed.

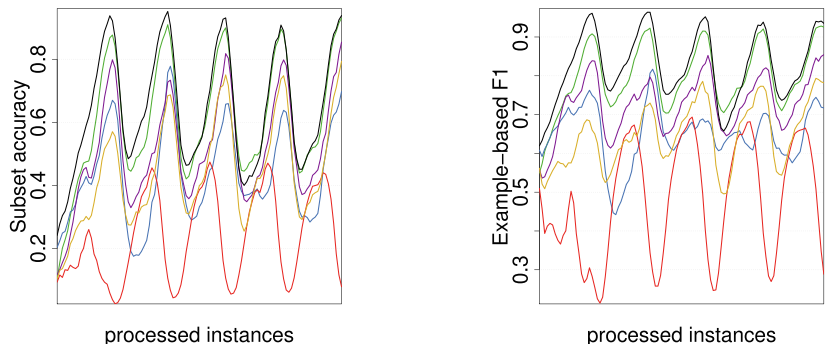


Fig. 10. Ensembles comparison: prequential subset accuracy and F1 on Yelp.

subspaces, no online bagging, and a single classifier. The performance for the Eukaryote and Human datasets are very similar for all the variants. Still, AESAKNNS shows the best overall subset accuracy and particularly F1, whereas the no online bagging and the single classifier variants exhibit a slightly worse performance. On the other hand, differences are significantly more remarkable for the Ohsumed and Scence datasets. There is a significant difference between the single classifier and the no online bagging model, especially for the Oshumed dataset. These results clearly show the advantages of the combination of the proposed strategies that altogether make AESAKNNS a robust ensemble.

Figs. 23,24 show the results of the Bayesian sign test of AESAKNNS vs each of the four variants without a key component, based on the subset accuracy and example-based F1, respectively. The test is very meaningful in reporting significant differences in the contribution of the online bagging and the ensemble vs the single classifier. On the other hand, the differences are smaller (the points in the figures are located closer to the rope - top of the triangle), yet in favor of AESAKNNS (located on the right hand side of the triangle) for the background ensemble and the feature subspaces. This means that there is room for improvement in the selection of the feature subspaces and the training of the background ensemble to build even more competitive ensembles.

5. Conclusions and future work

This paper introduced AESAKNNS, an adaptive ensemble of self-adjusting nearest neighbor subspaces for multi-label data streams. The ensemble employed the MLSAkNN base classifier to naturally adapt to concept drift. We proposed adaptive random subspaces and online bagging to increase the diversity of the base classifiers

Table 6
Subset accuracy for all nearest neighbor classifiers on each dataset.

Dataset	kNN	kNNP	kNNPA	SAMkNN	MLkNN	MLSAMkNN	MLSAMPkNN	MLSAkNN	AESAKNNS
Birds	0.479	**0.481**	**0.481**	0.468	0.473	0.475	0.468	0.454	0.454
Virus	0.483	0.468	0.468	0.477	0.533	0.553	0.566	0.632	**0.671**
Flags	0.116	0.109	0.109	0.041	0.087	0.075	0.068	**0.130**	0.123
Scene	0.544	0.649	0.648	0.679	0.529	0.828	0.846	0.846	**0.874**
Enron	0.089	0.095	0.086	0.087	0.072	0.098	**0.101**	0.095	0.095
Genbase	0.762	0.750	0.750	0.209	0.754	0.883	0.879	0.912	**0.918**
Medical	0.328	0.315	0.315	0.243	0.370	0.381	**0.418**	0.366	0.353
Water-qual	0.013	0.018	0.017	0.020	0.013	0.016	0.021	**0.057**	0.026
Corel-5 k	0.004	0.001	0.005	0.004	0.005	0.022	0.035	0.046	**0.055**
Eukaryote	0.343	0.284	0.448	0.625	0.299	0.698	0.717	0.739	**0.813**
Plant	0.147	0.155	0.161	0.332	0.192	0.417	0.566	0.616	**0.762**
Reuters	0.180	0.189	0.189	0.136	0.166	0.191	**0.249**	0.239	0.240
Mediamill	0.102	0.107	0.105	0.146	0.087	0.145	0.160	0.152	**0.189**
Ohsumed	0.007	0.005	0.005	0.011	0.014	0.026	0.032	0.033	**0.680**
CAL-500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Yelp	0.199	0.226	0.407	0.517	0.153	0.503	0.567	0.625	**0.665**
Slashdot	0.140	0.139	0.139	0.066	0.069	0.189	**0.205**	0.189	0.143
Human	0.197	0.197	0.328	0.498	0.119	0.579	0.645	0.668	**0.751**
Langlog	0.179	0.179	0.179	0.148	0.147	0.159	**0.188**	0.176	0.178
Gnegative	0.521	0.534	0.592	0.726	0.579	0.754	0.823	0.842	**0.896**
CHD	0.148	0.166	0.166	0.120	0.112	0.153	0.140	**0.231**	0.229
Stackex	0.011	0.007	0.007	0.013	0.007	0.010	0.014	0.013	**0.014**
Corel-16 k	0.011	0.006	0.020	0.024	0.009	0.070	0.087	0.117	**0.122**
Imdb	0.034	0.031	0.031	0.017	0.003	0.045	0.066	0.072	**0.093**
Nuswide-C	0.229	0.236	0.238	**0.266**	0.232	0.255	0.247	0.251	0.240
Nuswide-B	0.157	0.162	0.157	0.259	0.222	**0.260**	0.242	0.249	0.239
Yahoo-soc	0.121	0.126	0.112	0.155	0.095	0.118	0.148	0.143	**0.185**
Eurlex-sm	0.070	0.087	0.083	0.072	0.057	0.043	0.130	**0.133**	0.118
Hypersphe	0.518	0.480	0.527	0.524	0.513	0.522	0.510	0.523	**0.529**
Hypercube	0.995	0.992	0.996	0.998	0.998	0.998	0.997	0.996	**0.999**
Average	0.238	0.240	0.259	0.263	0.230	0.315	0.338	0.351	**0.389**
Rank	6.47	6.50	6.13	5.70	7.17	4.37	3.33	3.00	**2.33**

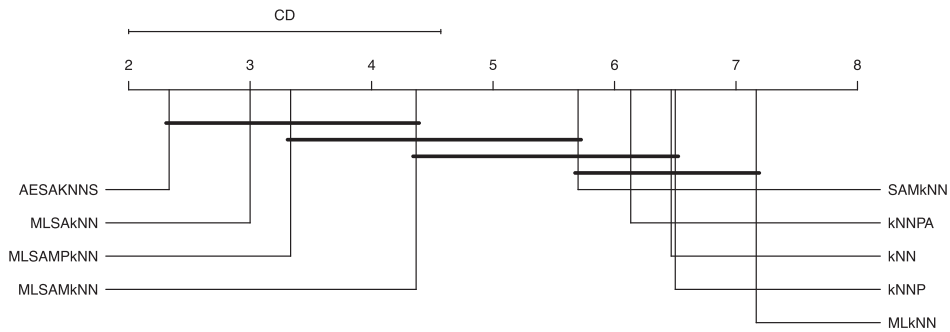


Fig. 11. Bonferroni-Dunn for subset accuracy on nearest neighbors.

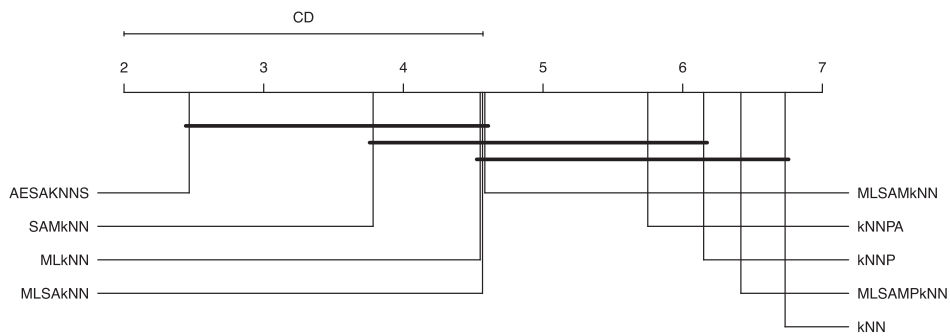


Fig. 12. Bonferroni-Dunn for Hamming score on nearest neighbors.

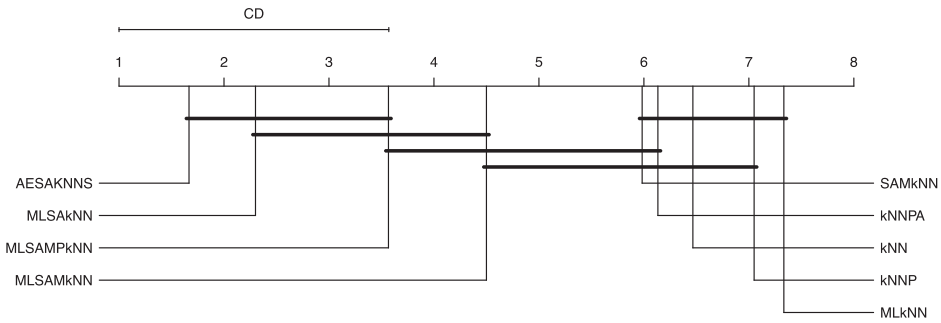
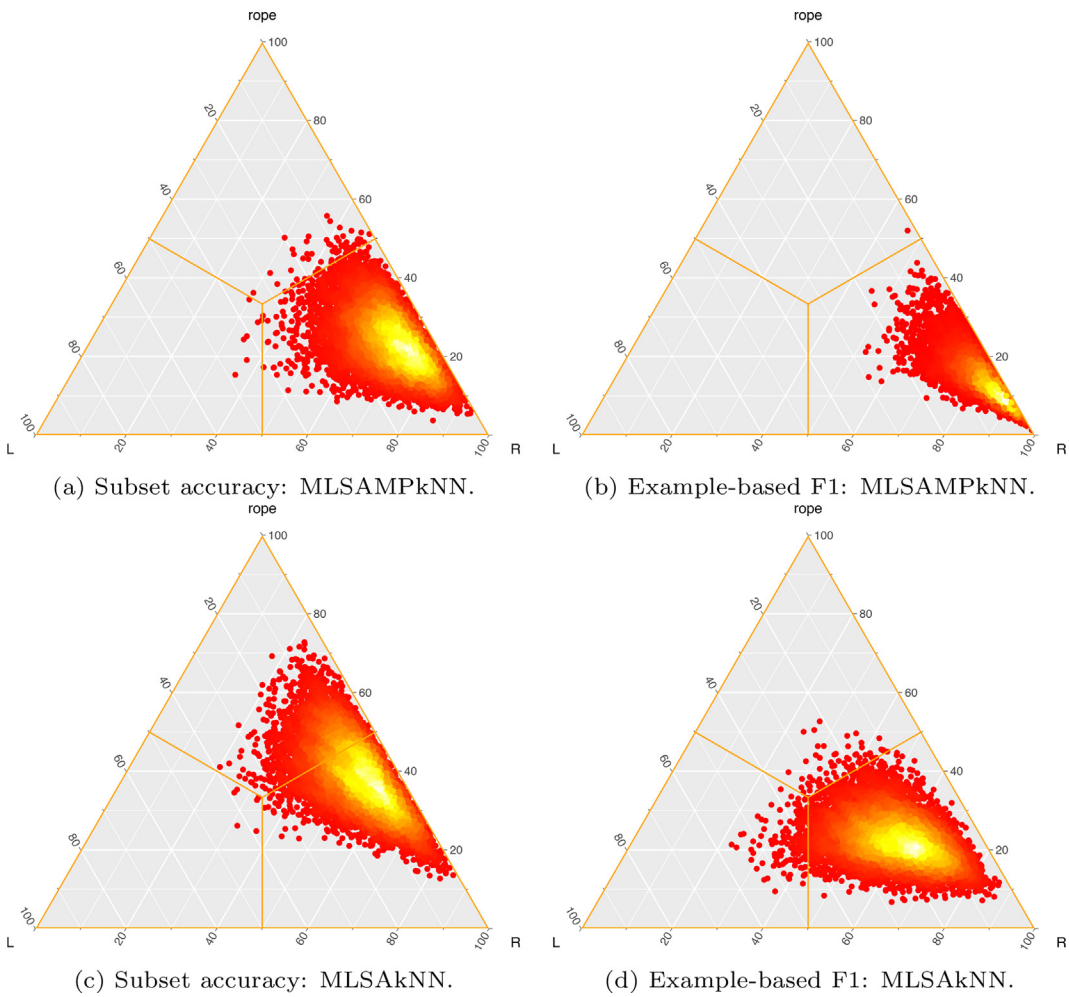


Fig. 13. Bonferroni-Dunn for example-based F1 on nearest neighbors.

Table 7
Wilcoxon signed test: AESAKNNS vs nearest neighbors (*p*-values).

Algorithm	Su. Acc	H. Sco	Ex. F1	Mi. F1	Ma. F1
kNN	3.33E-06	1.03E-04	1.44E-06	1.59E-06	8.67E-07
kNNP	4.03E-06	1.16E-04	1.18E-06	1.18E-06	8.67E-07
kNNPA	3.33E-06	1.31E-04	1.30E-06	1.30E-06	8.67E-07
SAMkNN	9.42E-06	4.99E-04	8.67E-07	1.44E-06	2.14E-06
MLkNN	4.44E-06	1.31E-04	1.06E-06	1.18E-06	9.60E-07
MLSAMkNN	1.36E-04	9.45E-05	4.23E-06	4.66E-06	4.66E-06
MLSAMPkNN	2.75E-03	1.76E-06	3.03E-05	1.86E-05	4.27E-05
MLSAkNN	1.54E-02	2.06E-03	2.36E-02	9.52E-03	4.55E-01



(a) Subset accuracy: MLSAMPkNN. (b) Example-based F1: MLSAMPkNN.

(c) Subset accuracy: MLSAkNN. (d) Example-based F1: MLSAkNN.

Fig. 14. Bayesian sign test on top nearest neighbors (AESAKNNS vs MLSAMPkNN and MLSAkNN).

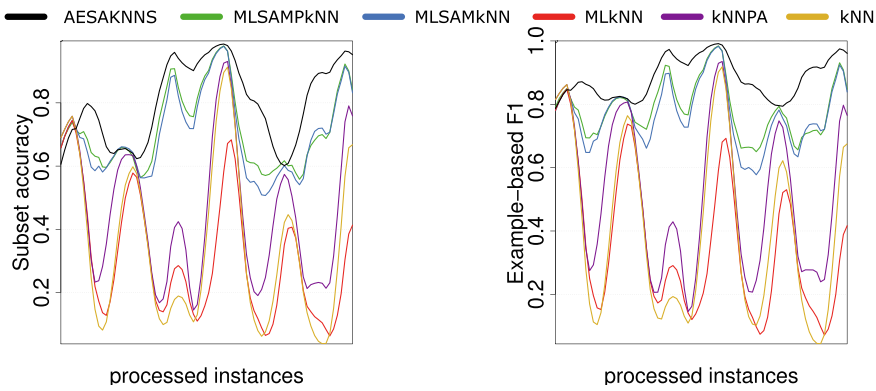


Fig. 15. kNN comparison: prequential subset accuracy and F1 on Eukaryote.

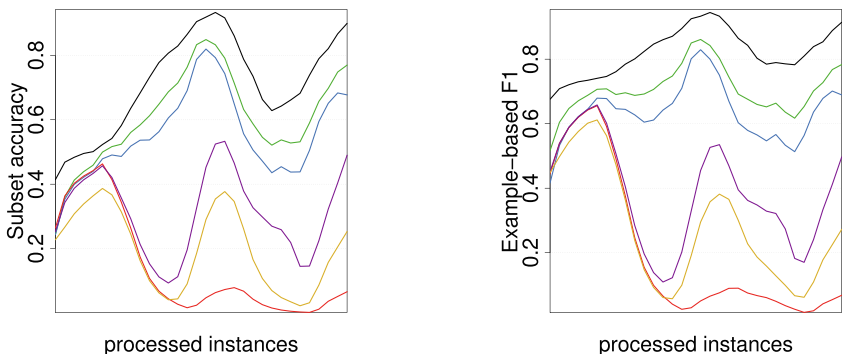


Fig. 16. kNN comparison: prequential subset accuracy and F1 on Human.

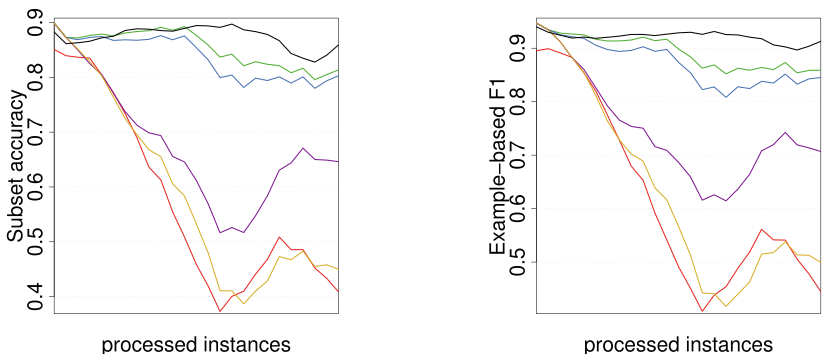


Fig. 17. kNN comparison: prequential subset accuracy and F1 on Scene.

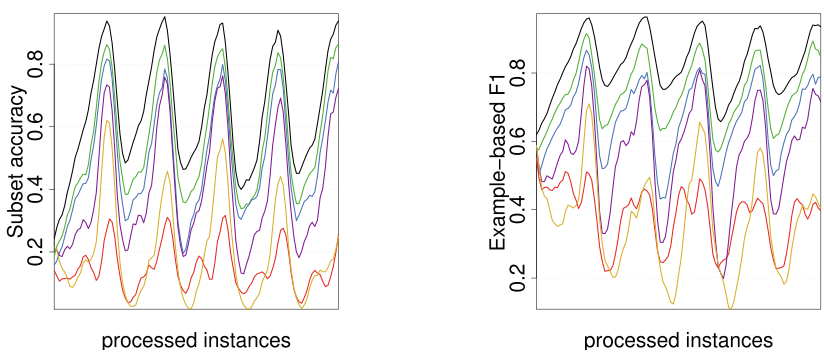


Fig. 18. kNN comparison: prequential subset accuracy and F1 on Yelp.

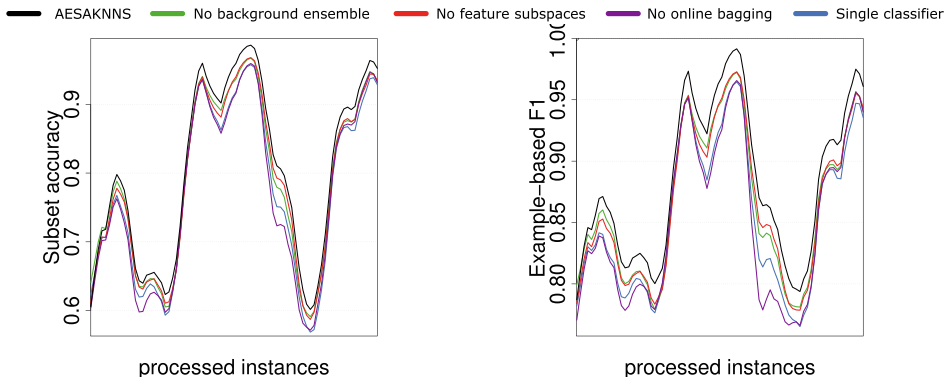


Fig. 19. AESAKNNS contributions: prequential subset accuracy and F1 on Eukaryote.

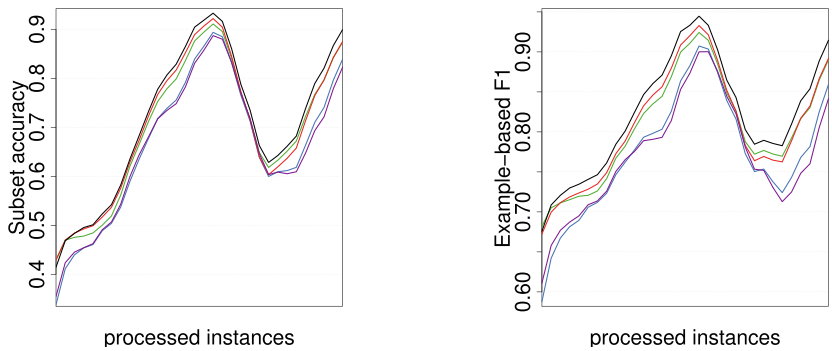


Fig. 20. AESAKNNS contributions: prequential subset accuracy and F1 on Human.

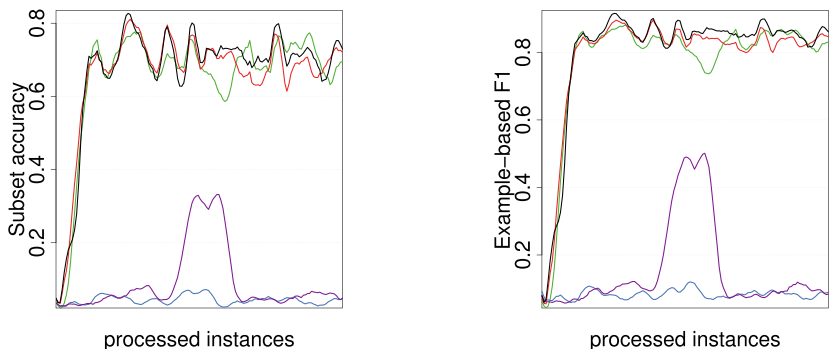


Fig. 21. AESAKNNS contributions: prequential subset accuracy and F1 on Ohsumed.

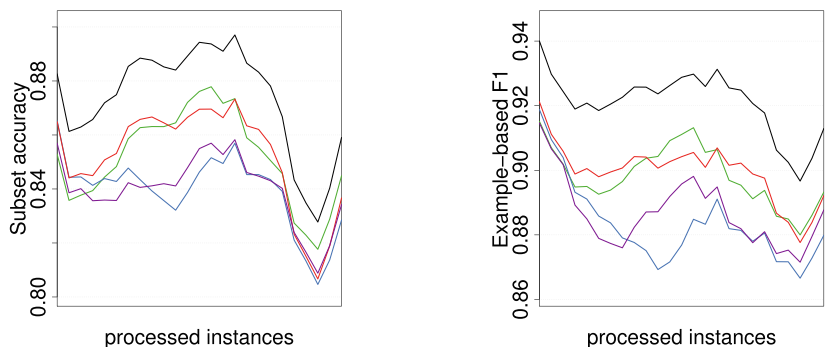
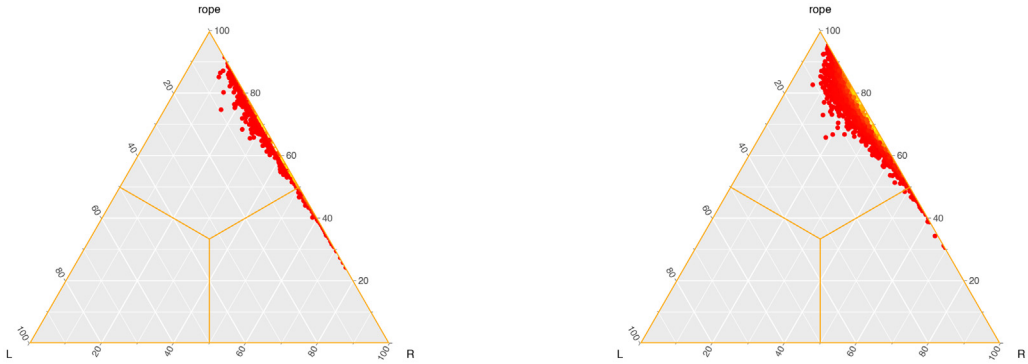
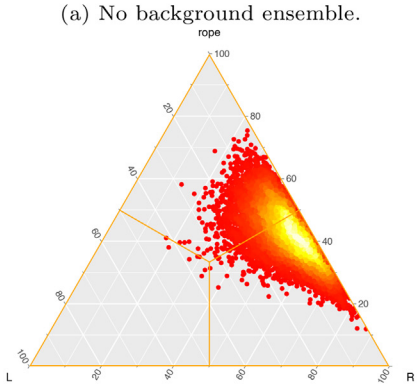


Fig. 22. AESAKNNS contributions: prequential subset accuracy and F1 on Scene.

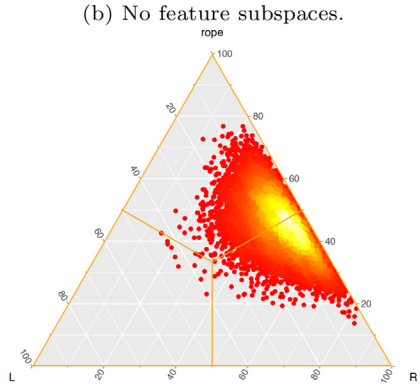


(a) No background ensemble.

(b) No feature subspaces.

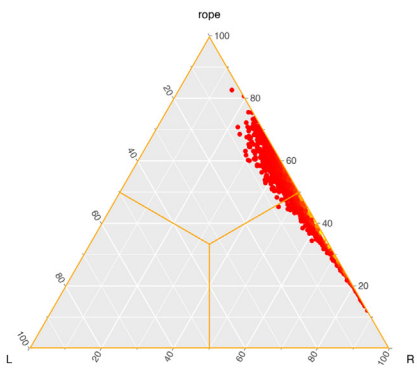


(c) No online bagging.

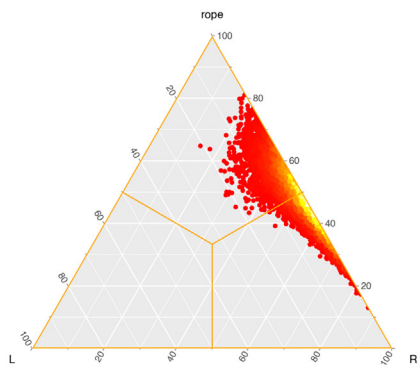


(d) Single classifier.

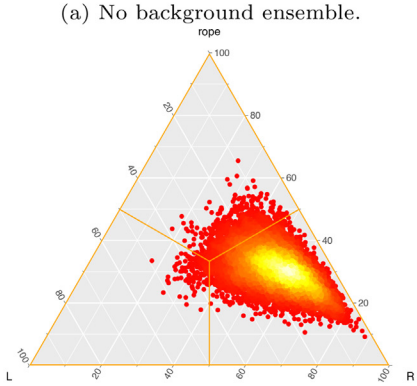
Fig. 23. Bayesian sign test: subset accuracy on AESAKNNS four main contributions.



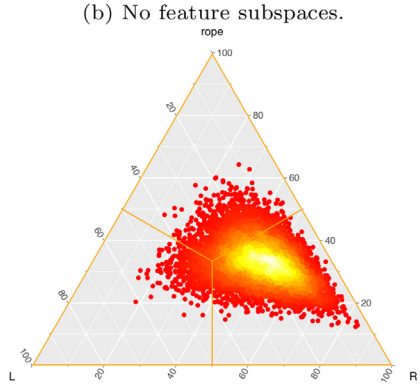
(a) No background ensemble.



(b) No feature subspaces.



(c) No online bagging.



(d) Single classifier.

Fig. 24. Bayesian sign test: example-based F1 on AESAKNNS four main contributions.

in the ensemble. We proposed to monitor concept drift on the subspaces using a collection of ADWIN detectors to build a background ensemble on new varying-size feature subspaces. These combined mechanisms allowed AESAKNNS to be highly adaptive to concept drift and overcome other various multi-label data difficulties. We presented a thorough experimental study to evaluate how competitive AESAKNNS is against 30 different models across 30 datasets and 12 multi-label metrics. Achieving top performance for 9 out of 12 metrics. We analyzed how AESAKNNS compares to other ensemble and nearest neighbor algorithms and confirmed through statistical analyses that AESAKNNS is well rounded classifier. Moreover, we provided a detailed analysis on the performance impact of each of the contributions of our method.

As future work, there are more adaptive mechanisms that could potentially benefit AESAKNNS. This could include adaptive windowing and additional hyperparameter adjustment, dynamic ensemble sizing to increase the amount of knowledge gained during meta evaluations, and combining feature, instance, and label subspaces to create more diverse and specialized base classifiers.

CRediT authorship contribution statement

Gavin Alberghini: Formal analysis, Investigation, Resources, Writing - original draft. **Sylvio Barbon Junior:** Resources, Writing - review & editing. **Alberto Cano:** Conceptualization, Methodology, Software, Supervision, Writing - original draft.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

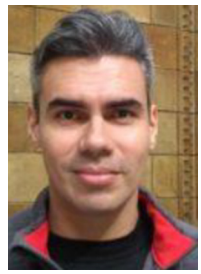
This research was partially supported by the 2018 VCU Presidential Research Quest Fund and an Amazon AWS Machine Learning Research award. High Performance Computing resources provided by the High Performance Research Computing (HPRC) Core Facility at Virginia Commonwealth University were used for conducting the research reported in this work.

References

[1] W. Liu, H. Wang, X. Shen, I. Tsang, The emerging trends of multi-label learning, IEEE Transactions on Pattern Analysis and Machine Intelligence (2021).
[2] J.M. Moyano, E.L. Gibaja, K.J. Cios, S. Ventura, Review of ensembles of multi-label classifiers: models, experimental study and prospects, Information Fusion 44 (2018) 33–45.
[3] V.-L. Nguyen, E. Hüllermeier, M. Rapp, E.L. Mencía, J. Fürnkranz, On aggregation in ensembles of multilabel classifiers, in: International Conference on Discovery Science, 2020, pp. 533–547..
[4] E. Gibaja, S. Ventura, Multi-label learning: a review of the state of the art and ongoing research, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 4 (6) (2014) 411–444.
[5] H.M. Gomes, A. Bifet, J. Read, J.P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, T. Abdessalem, Adaptive random forests for evolving data stream classification, Machine Learning 106 (9) (2017) 1469–1495.
[6] I.A. Lawal, S.A. Abdulkarim, Adaptive SVM for data stream classification, South African Computer Journal 29 (1) (2017) 27–42.
[7] B. Krawczyk, A. Cano, Online ensemble learning with abstaining classifiers for drifting and noisy data streams, Applied Soft Computing 68 (2018) 677–692.
[8] A. Cano, B. Krawczyk, Kappa Updated Ensemble for Drifting Data Stream Mining, Machine Learning 109 (1) (2020) 175–218.
[9] B. Krawczyk, L.L. Minku, J. Gama, J. Stefanowski, M. Woźniak, Ensemble learning for data stream analysis: A survey, Information Fusion 37 (2017) 132–156.
[10] Y. Sun, K. Tang, L.L. Minku, S. Wang, X. Yao, Online ensemble learning of data streams with gradually evolved classes, IEEE Transactions on Knowledge and Data Engineering 28 (6) (2016) 1532–1545.
[11] M. Roseberry, B. Krawczyk, Y. Djenouri, A. Cano, Self-adjusting k nearest neighbors for continual learning from multi-label drifting data streams, Neurocomputing 442 (2021) 10–25.
[12] M.-L. Zhang, Y.-K. Li, X.-Y. Liu, X. Geng, Binary relevance for multi-label learning: an overview, Frontiers of Computer Science 12 (2) (2018) 191–202.
[13] J. Liu, Y. Li, W. Weng, J. Zhang, B. Chen, S. Wu, Feature selection for multi-label learning with streaming label, Neurocomputing 387 (2020) 268–278.
[14] M.-L. Zhang, Z.-H. Zhou, ML-KNN: A lazy learning approach to multi-label learning, Pattern recognition 40 (7) (2007) 2038–2048.
[15] N.C. Oza, S.J. Russell, Online bagging and boosting, in: International Workshop on Artificial Intelligence and Statistics, 2001, pp. 229–236..
[16] J. Gama, P. Kosina, Recurrent concepts in data streams classification, Knowledge and Information Systems 40 (3) (2014) 489–507.
[17] D. You, Y. Wang, J. Xiao, Y. Lin, M. Pan, Z. Chen, L. Shen, X. Wu, Online multi-label streaming feature selection with label correlation, IEEE Transactions on Knowledge and Data Engineering (2021).
[18] A. Fernández, S. Garcia, F. Herrera, N.V. Chawla, SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary, Journal of Artificial Intelligence Research 61 (2018) 863–905.
[19] C. Drummond, R.C. Holte, et al., C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling, in: Workshop on learning from imbalanced datasets, Vol. 11, 2003, pp. 1–8..
[20] A. Tarekegn, M. Giacobini, K. Michalak, A review of methods for imbalanced multi-label classification, Pattern Recognition 107965 (2021).
[21] M. Roseberry, A. Cano, Multi-label knn classifier with self adjusting memory for drifting data streams, in: International Workshop on Learning with Imbalanced Domains: Theory and Applications, 2018, pp. 23–37..
[22] M. Roseberry, B. Krawczyk, A. Cano, Multi-label punitive knn with self-adjusting memory for drifting data streams, ACM Transactions on Knowledge Discovery from Data 13 (6) (2019) 1–31.
[23] T. Museba, F. Nelwamondo, K. Ouahada, An adaptive heterogeneous online learning ensemble classifier for nonstationary environments, Computational Intelligence and Neuroscience 2021 (2021).
[24] E. Montanes, R. Senge, J. Barranquero, J.R. Quevedo, J.J. del Coz, E. Hüllermeier, Dependent binary relevance models for multi-label classification, Pattern Recognition 47 (3) (2014) 1494–1508.
[25] A. Rivolli, J. Read, C. Soares, B. Pfahringer, A.C. de Carvalho, An empirical analysis of binary transformation strategies and base algorithms for multi-label learning, Machine Learning 109 (8) (2020) 1509–1563.
[26] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, Machine learning 85 (3) (2011) 333.
[27] L. Rokach, A. Schclar, E. Itach, Ensemble methods for multi-label classification, Expert Systems with Applications 41 (16) (2014) 7507–7523.
[28] E.C. Gatto, M. Ferrandin, R. Cerri, Exploring label correlations for partitioning the label space in multi-label classification, in, International Joint Conference on Neural Networks (IJCNN) (2021) 1–8.
[29] R. Wang, S. Kwong, X. Wang, Y. Jia, Active k-labelsets ensemble for multi-label classification, Pattern Recognition 109 (2021) 107583.
[30] R. Senge, J.J. del Coz, E. Hüllermeier, Rectifying classifier chains for multi-label classification, arXiv preprint arXiv:1906.02915 (2019)..
[31] M.-L. Zhang, Y.-K. Li, H. Yang, X.-Y. Liu, Towards class-imbalance aware multi-label learning, IEEE Transactions on, Cybernetics (2020).
[32] Q. Wu, M. Tan, H. Song, J. Chen, M.K. Ng, ML-FOREST: A multi-label tree ensemble method for multi-label classification, IEEE transactions on knowledge and data engineering 28 (10) (2016) 2665–2680.
[33] T. Wei, J.-X. Shi, Y.-F. Li, Probabilistic label tree for streaming multi-label learning, in, in: 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2021, pp. 1801–1811.
[34] J. Huang, F. Qin, X. Zheng, Z. Cheng, Z. Yuan, W. Zhang, Q. Huang, Improving multi-label classification with missing labels by learning label-specific features, Information Sciences 492 (2019) 124–146.
[35] C. Zhang, Z. Li, Multi-label learning with label-specific features via weighting and label entropy guided clustering ensemble, Neurocomputing 419 (2021) 59–69.
[36] X. Zheng, P. Li, Z. Chu, X. Hu, A survey on multi-label data stream classification, IEEE Access 8 (2019) 1249–1275.
[37] H. Zhang, W. Liu, S. Wang, J. Shan, Q. Liu, Resample-based ensemble framework for drifting imbalanced data streams, IEEE Access 7 (2019) 65103–65115.
[38] Y. Sun, H. Shao, S. Wang, Efficient ensemble classification for multi-label data streams with concept drift, Information 10 (5) (2019) 158.
[39] R. Sousa, J. Gama, Multi-label classification from high-speed data streams with adaptive model rules and random rules, Progress in Artificial Intelligence 7 (3) (2018) 177–187.
[40] J.C. Júnior, E. Faria, J. Silva, R. Cerri, Label powerset for multi-label data streams classification with concept drift, in: in: 5th Symposium on Knowledge Discovery, Mining and Learning, 2017, pp. 97–104.
[41] J.D.C. Júnior, E.R. Faria, J.A. Silva, J. Gama, R. Cerri, Pruned sets for multi-label stream classification without true labels, in: International Joint Conference on Neural Networks, 2019, pp. 1–8..
[42] J.D.C. Júnior, E.R. Faria, J.A. Silva, J. Gama, R. Cerri, Novelty detection for multi-label stream classification, 8th Brazilian Conference on Intelligent Systems (2019) 144–149.
[43] R. Cerri, J.D.C. Júnior, E.R. d. F. Paiva, J.M.P. da Gama, Multi-label stream classification with self-organizing maps, arXiv preprint arXiv:2004.09397 (2020)..

[44] R. Cerri, J.D.C. Júnior, E.R. Faria, J. Gama, A new self-organizing map based algorithm for multi-label stream classification, in: 36th Annual ACM Symposium on Applied Computing, 2021, pp. 418–426.
[45] A. Bifet, G. Holmes, B. Pfahringer, Leveraging bagging for evolving data streams, in: European Conference on Machine Learning and Knowledge Discovery in Databases, 2010, pp. 135–150.
[46] A. Bifet, R. Gavalda, Learning from time-changing data with adaptive windowing, in: SIAM international conference on data mining, 2007, pp. 443–448.
[47] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: massive online analysis, Journal of Machine Learning Research 11 (2010) 1601–1604.
[48] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, R. Gavalda, New ensemble methods for evolving data streams, in: ACM SIGKDD international conference on Knowledge discovery and data mining, 2009, pp. 139–148.
[49] P. Domingos, G. Hulten, Mining high-speed data streams, in: ACM SIGKDD international conference on Knowledge discovery and data mining, 2000, pp. 71–80.
[50] B. Pfahringer, G. Holmes, R. Kirkby, New options for hoeffding trees, in: Australasian Joint Conference on Artificial Intelligence, 2007, pp. 90–99..
[51] M. Baena-Garcia, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales-Bueno, Early drift detection method, in: International Workshop on Knowledge Discovery from Data Streams, Vol. 6, 2006, pp. 77–86..
[52] R. Pelossof, M. Jones, I. Vovsha, C. Rudin, Online coordinate boosting, in: International Conference on Computer Vision Workshops, 2009, pp. 1354–1361.
[53] J.Z. Kolter, M.A. Maloof, Dynamic weighted majority: An ensemble method for drifting concepts, Journal of Machine Learning Research 8 (2007) 2755–2790.
[54] D. Brzezinski, J. Stefanowski, Reacting to different types of concept drift: The accuracy updated ensemble algorithm, IEEE Transactions on Neural Networks and Learning Systems 25 (1) (2013) 81–94.
[55] P. Kranen, H. Kremer, T. Jansen, T. Seidl, A. Bifet, G. Holmes, B. Pfahringer, J. Read, Stream data mining using the moa framework, in: International Conference on Database Systems for Advanced Applications, 2012, pp. 309–313.
[56] A. Bifet, B. Pfahringer, J. Read, G. Holmes, Efficient data stream classification via probabilistic adaptive windows, in: ACM symposium on applied computing, 2013, pp. 801–806.
[57] V. Losing, B. Hammer, H. Wersing, Knn classifier with self adjusting memory for heterogeneous concept drift, in: IEEE International Conference on Data Mining, 2016, pp. 291–300.
[58] J. Read, P. Reutemann, B. Pfahringer, G. Holmes, MEKA: A multi-label/multi-target extension to Weka, Journal of Machine Learning Research 17 (21) (2016) 1–5.
[59] J. Read, A. Bifet, G. Holmes, B. Pfahringer, Streaming multi-label classification, in: Workshop on Applications of Pattern Analysis, 2011, pp. 19–25..
[60] R. Sousa, J. Gama, Online multi-label classification with adaptive model rules, in: Conference of the Spanish Association for Artificial Intelligence, 2016, pp. 58–67.
[61] J. Read, A. Bifet, G. Holmes, B. Pfahringer, Scalable and efficient multi-label classification for evolving data streams, Machine Learning 88 (1–2) (2012) 243–272.
[62] J. Gama, P.P. Rodrigues, R. Sebastiao, Evaluating algorithms that learn from data streams, in: ACM symposium on Applied Computing, 2009, pp. 1496–1500..

Gavin Alberghini is a software application development engineer at MITRE. He received the M.Sc. degree in Computer Science and B.Sc. degree in Computer Science from the Virginia Commonwealth University, USA in 2021 and 2020, respectively. His interests are data science and machine learning.



Sylvio Barbon Junior, PhD, is Associate Professor at Department of Engineering and Architecture at University of Trieste (UNITS), Italy. He received his BSc degree in Computer Science in 2005, his MSc degree in Computational Physics from the University of Sao Paulo (2007), a degree in Computational Engineering in 2008, and a PhD degree (2011) from IFSC/USP similar to his MSc degree. In 2017, he was a visiting researcher at the University of Modena and Reggio Emilia (Italy), working on multispectral analysis and machine learning. In 2021, as visiting professor at the Universita Degli Studi Di Milano (Italy), he focused on data stream and process mining. He is currently a professor in postgraduate and graduate programs. His research interests include digital signal processing, pattern recognition, and machine learning.

Alberto Cano, PhD, is an Associate Professor with the Department of Computer Science, Virginia Commonwealth University, USA, where he heads the High-Performance Data Mining Lab. He obtained his BSc degrees in Computer Engineering and in Computer Science from the University of Cordoba, Spain, in 2008 and 2010, respectively, and his MSc and PhD degrees in Intelligent Systems and Computer Science from the University of Granada, Spain, in 2011 and 2014 respectively. His research is focused on machine learning, data mining, general-purpose computing on graphics processing units, Apache Spark, and evolutionary computation. He has published over 50 articles in high-impact factor journals, 50 contributions to international conferences, two book chapters, and one book in the areas of machine learning, data mining, and parallel, distributed, and GPU computing. His research is supported by an Amazon AWS Machine Learning award and the VCU Presidential Research Quest Fund. Dr. Cano is Associate Editor of IEEE Access and PeerJ Computer Science.