Virginia Commonwealth University
VCU Scholars Compass

Theses and Dissertations | Graduate School

2021

CONTINUAL LEARNING FOR MULTI-LABEL DRIFTING DATA STREAMS USING HOMOGENEOUS ENSEMBLE OF SELF-ADJUSTING NEAREST NEIGHBORS

Gavin Alberghini
Virginia Commonwealth University

Follow this and additional works at: https://scholarscompass.vcu.edu/etd

 Part of the Data Science Commons

© The Author

Downloaded from
https://scholarscompass.vcu.edu/etd/6562

This Thesis is brought to you for free and open access by the Graduate School at VCU Scholars Compass. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of VCU Scholars Compass. For more information, please contact libcompass@vcu.edu.

©Gavin Alberghini, May 2021

All Rights Reserved.

THESIS: CONTINUAL LEARNING FOR MULTI-LABEL DRIFTING DATA STREAMS USING HOMOGENEOUS ENSEMBLE OF SELF-ADJUSTING NEAREST NEIGHBORS

A Thesis: submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

by

GAVIN ALBERGHINI

Bachelor of Science, Virginia Commonwealth University, 2017 - 2020

Director: Thesis: Alberto Cano,

Assistant Professor, Department of Computer Science

Virginia Commonwealth University

Richmond, Virginia

May, 2021

Acknowledgements

I would like to thank my parents, Leah and John, for all of the love and support they have given me as I work toward my dreams. I would also like to thank Dr. Cano, who has been an excellent instructor and mentor to me over the years. His assistance on this project and involvement in my intellectual growth will always be remembered and appreciated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

THESIS: CONTINUAL LEARNING FOR MULTI-LABEL DRIFTING DATA STREAMS USING HOMOGENEOUS ENSEMBLE OF SELF-ADJUSTING NEAREST NEIGHBORS

By Gavin Alberghini

A Thesis: submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

Virginia Commonwealth University, 2021.

Director: Thesis: Alberto Cano,
Assistant Professor, Department of Computer Science

Multi-label data streams are sequences of multi-label instances arriving over time to a multi-label classifier. The properties of the data stream may continuously change due to concept drift. Therefore, algorithms must constantly adapt to the new data distributions. In this thesis a novel ensemble method for multi-label drifting streams named Homogeneous Ensemble of Self-Adjusting Nearest Neighbors (HESAkNN) is proposed. 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. Upon detection, the algorithm automatically trains additional classifiers in the background to attempt to capture

new concepts. After a pre-determined number of instances, both active and background classifiers are compared and only the most accurate classifiers are selected to populate the new active ensemble. The 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 11 performance metrics. Results validated using non-parametric statistical analysis support the better performance of the HESAkNN and highlight the contribution of its components in improving the performance of the ensemble.

CHAPTER 1

INTRODUCTION

1.1 Background

As industry attempts to leverage machine learning to solve more advanced problems, multi-label data is becoming more readily available and necessary for knowledge discovery [1]. Some application domains include image labeling, road monitoring, and social network mining [2, 3, 4]. Historically, multi-label algorithms have been time and memory consuming, making them infeasible for real-time systems. However, as the need for real-time analysis of large and complex data increases, multi-label has been identified as an effective method for processing big data. Multi-label data streams merges two challenging tasks: multi-label classification and data stream mining. Multi-label classification is a generalization of the multi-class classification problem. In multi-label classification, each example's class contains multiple, non-exclusive labels instead of a singular class value. While multi-label classification has experienced significant research contributions over the last few years, the context of online learning remains an open issue. Data stream classification is a trending area of research [5, 6]. Data contained in real-world applications are 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) while maintaining a high accuracy and robustness to concept drift. The former issue of time and memory utilization is not unique to this problem space. Batch learning still hold

these values in high regard, they are however, not necessary for all issues that are described as batch learning problems. Online learning is based on the idea of real time feedback and learning instance by instance. These concepts of how instances correlate to training and prediction time introduce 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. Consequently, algorithms must adapt and adjust to address concept drift in a continual learning environment. Ensembles are popular approaches for concept drift since the algorithm may add/remove classifiers to add/forget concepts appearing/fading in the stream. However, there are very limited studies on the use of ensembles for multi-label data streams. Moreover, most of the ensembles for data streams are based on Hoeffding Trees, but nearest neighbor models have demonstrated superior performance as a base model for multi-label classification.

This thesis introduces a novel ensemble algorithm for multi-label data streams named Homogeneous Ensemble of Self-Adjusting Nearest Neighbors (HESAkNN). It is based on the idea of ensembles of classifiers on feature and instance subspaces proposed in KUE [7] and the self-adjusting algorithm MLSAkNN [8] as base classifier. HESAkNN 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 the online learning environment.

1.2 Contributions

- HESAkNN: a robust homogeneous ensemble for self-adjusting to concept drift using multi-label nearest neighbors.
- A methodology to increase the ensemble diversity by combining feature sub-

spaces and online bagging.

- A self-adjusting nearest neighbor classifier as base model for the ensemble.
- A background ensemble to adapt to concept drift upon detection of a warning using ADWIN.
- A thorough experimental study comparing HESAkNN to other state of the art models, including a more in depth analysis of the ensemble mechanisms and how they contribute to improve the classification performance.

CHAPTER 2

LITERATURE REVIEW

2.1 Multi-label stream classification

A multi-label stream is a potentially unbounded sequence $< S_1, S_2, ..., S_n, ... >$, 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 the labelset, a set of labels simultaneously associated with the instance. Multi-label classification can be viewed as a generalization of multi-class classification. 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 - AA), or transforming data into separate problems that can be solved via more traditional methods (Problem Transformation). A brief description of these two methods along with other problems in the multi-label streaming domain are listed below:

Problem transformation is centered on the idea of changing multi-label examples into problems that can be solved using already established methods. This is commonly used for creating base-line methods for multi-label data.

Label Combination: Label combination (LC), also known as label powerset (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: 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 [9] stemming from the loss of any label correlation when the problem is decomposed. However, multiple mechanisms exist to combat this issue [9].

Classifier Chains: 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 as problem transformation and focus on making changes to decision functions allowing a previously single class model to now operate in the multi-label context, including methods for feature selection [10]. Some examples include MLkNN [11] and MMP [12]. For this thesis, several adaptations of the kNN algorithm are considered for the multi-label context. In the work by Roseberry et al. [13] they proposed MLSAMkNN, a multi-label kNN for data stream mining that introduced the first iteration of concept drift mechanisms for the MLkNN classifier using a self-adjusting memory for the sliding window. In their following work [14], they propose MLSAMPkNN, an improvement by adding a punitive system that can selectively remove instances to improve classifier performance. Finally, in [8], the authors proposed MLSAkNN, a self-adjusting k value that dynamically changes over time for each label, adapting automatically to the best parameter settings in real-time.

2.1.1 Concept drift

Concept drift is the term used to describe how the statistical properties of target labelset y change over time. Formally concept drift is defined 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. When the underlying associations in the data change, models built on past data are no longer effective. 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 some target distribution. Models can 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 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 [15]. 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.

2.1.2 Class imbalance in multi-label streams

Class imbalance is one of the most critical issues to understand for multi-class and multi-label learning. Many supervised learning algorithms rely on the assumption of equal class distributions. By skewing this proportion, one can expect to see poor predictive performance for the minority class. There are several methods to combat class imbalance, including random sampling methods such as over-sampling and under-sampling [16, 17]. In the multi-label learning context, one can consider the class imbalance of each label (following the BR approach) or the label set (following the LP approach). 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. Salazar et al. [18] performed experiments focusing on the use of data augmentation for semi-supervised learning with credit card frauds data. They discover relationships between semi-supervised algorithms and data augmentation ratios, along with the discovery of key benchmarks for business making decisions. In the work of Zhang et al. [19], the authors drive to design a learning strategy that explores both label correlations and class imbalance simultaneously. As we move into the data stream context the difficulties of class imbalance increase. Detailed in the work of Zheng et al. [20], class imbalance is a significant problem for data streams. They discuss how many existing methods for combating class imbalance rely on static proportions of imbalance over time, which is unrealistic for practical applications. In

addition, they recommend studying change in multiple minority class imbalance in order to benefit future analysis. The important consideration is that throughout a stream data proportions are constantly shifting. Accompanying the dynamic problems of concept drift, ideally classifiers should be able to incorporate mechanisms to handle adaptive class imbalance. The continued work of Zhang et al. [21] depicts a re-sampling ensemble framework to combat this issue. Roseberry et al. [8] also employed the self-adjusting k value to better adapt to underrepresented minority labels, adding a degree of freedom for label-independent k neighbors based on their best self-adjusting parameters. This follows the idea of Zhang and Wu in creating label-specific multi-label classifiers [22].

2.1.3 Ensemble learning for multi-label streaming scenarios

Ensemble learning is a popular methodology of classification in machine learning. Ensembles are collections of classifiers that represent different 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 thesis, the focus falls 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.

2.1.3.1 Ensembles for data streams

Ensembles are among the most robust and accurate learners available for complex machine learning tasks [23, 24]. Over recent years of research, ensemble learning has been identified as an effective method for data stream mining. Some earlier methods

in this context include Oza et al. [25] and their work with boosting and bagging methods for improved ensemble performance. Adaptive Random Forests [5] is another more traditional ensemble method with high popularity. This algorithm focuses on adaptive example re-sampling along with additional flexible operators. In the work of Sun et al. [26] 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. 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 algorithm dynamically enables and disables component classifiers based on the currently detected concepts, allowing the ensemble to adapt to the data. Museba et al. [27] 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. The drift detection method (DDM) is also utilized to detect data drift. On 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 of kappa and classifier voting abstinence as a means to combat concept drift is the algorithm proposed in Cano et al. [7, 28]. The important aspect of these 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 are better performers in the data stream mining context, because they are able to adjust to non-stationary data dynamically.

2.1.3.2 Ensembles for multi-label data

As touched on previously, there are two major types of methods for dealing with multi-label data, problem transformation, and algorithm adaptation. Ensembles have been widely used as a method for implementing binary relevance, label-powerset, or classifier chains in various ways [29, 30, 31, 32]. When used for BR they commonly are created as n length ensembles that contain one traditional binary classifier per label. In the LP problem domain, ensembles are typically used to distribute the large class space among multiple classifiers. Finally, when used for CC, ensembles typically express different chain orders to improve the chance of discovering the true label dependency. However, ensembles for algorithm adaption methods are popular as well [33, 34] and can provide similar benefits as seen for traditional ensemble methods. Many of these different ensembles exist for multi-label classification. In the work of Wu et al. [35], 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. These trees are then combined into an ensemble that forms compounded predictions based on how each tree structure is modeled. The ideas proposed by Huang et al. [36] 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 [37] and demonstrated that individual labels have their own representative features, this idea helps to guide meta-based ensemble methods to better understand relevant data for individual labels in an example.

When dealing with multi-label data streams, we must understand the issues of

both multi-label data and data streams. In recent literature, there is an emergence of methods targeting this domain. Sun et al. [38] establish in their work that most algorithms built for multi-label data streams are expansions of single-label classification streams, meaning almost all of these algorithms entirely ignoring potential label dependencies. In their work, they propose an ensemble algorithm supported by Jensen-Shannon concept drift detection. Their algorithm leverages infrequent label pruning as a method to improve classification performance by exploiting the label dependencies. 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. 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. The work of Wu et al. [40] explores the traditional random forest algorithm, creating an adaptive method for multi-label data streams based on adaptive random forest. This method was then implemented over the Hadoop framework to be competitive against other high speed algorithms in this context. The proposed algorithm HESAkNN is designed to meet the challenges of this problem domain via adaptive, multi-label component classifiers and intelligently designed ensemble methods.

CHAPTER 3

METHODOLOGY

3.1 HESAkNN

This section of the methodology presents Homogeneous Ensemble of Self-Adjusting Nearest Neighbors (HESAkNN) as a front running algorithm for multi-label data stream mining. The pseudo-code is presented in Alg. 1. As shown in the previous sections, there are many advantages to ensemble learning for both the data streaming and multi-label problem domains. HESAkNN leverages Multi-Label Self-Adapting kNN (MLSAkNN) [8] algorithms that are individually capable of adjusting for concept drift and other changes in multi-label data. 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 information on the problem. HESAkNN 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 has been detected. On detection the ensemble immediately resets the learning on active classifiers, to potentially capture the new concept with existing feature spaces. However, after drift detection, HESAkNN will 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, HESAkNN analyzes the Hamming score and subset accuracy of all classifiers from both ensembles. The best performing 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. Each module of HESAkNN will be described in the following subsections.

3.1.1 MLSAkNN

MLSAkNN is a multi-label algorithm designed by Roseberry et al. [8]. 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. Our contribution is to add another level of abstraction by creating ensembles of MLSAkNN classifiers where instances presented to the classifiers are subject to feature subspace projections and online bagging. This way, the diversity of the classifiers both in the feature and instance space is increased, leading to better predictions than individual classifiers.

Algorithm 1: HESAkNN algorithm.

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
$\hat{y}$: label set prediction of a base classifier

for $S_i \in \{S_1, ..., S_n\}$ **do**
  if S_1 **then**
    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
  $\hat{y} \leftarrow E$ prediction on S_i
  $\alpha \leftarrow$ ADWIN update on S_i and $\hat{y}$
  if α *warning for any* $\gamma \in E$ **then**
    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**
    $k \leftarrow Poisson(1)$ 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**
      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**
      $h'_j \leftarrow$ 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**
      $E \leftarrow Select(E \cup E', h \cdot s, h' \cdot s', m)$
      $E' \leftarrow \emptyset$
    end
  end
end

3.1.2 Online bagging

Bagging is the process of training a set of weak learners in parallel to unique collections of information. In this context, HESAkNN utilizes 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 of boosting applies additional weight to instances for specific learners to derive new knowledge, allowing us to sample with replacement from our data stream. Online bagging has been successfully used in Leveraging Bagging [41], Adaptive Random Forest [5], and Kappa Updated Ensemble [7].

3.1.3 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, but not all of the attributes are relevant for all of the labels. HESAkNN leverages the diversity of each classifier to solve this issue. Each weak classifier within the ensemble is given a unique subset of features to consider. During evaluations of the ensemble, classifiers with weak performance are discarded in favor of others with better performance. Because new classifiers are constantly being created and evaluated throughout the duration of the data stream, HESAkNN 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. The subspace size is modeled using a normal distribution with a mean 70% of the number of features.

3.1.4 Concept drift detection using ADWIN

ADWIN is an adaptive windowing algorithm [42] 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, 42, 41]. In HESAkNN, ADWIN is used as the primary source of explicit drift detection on each of the base classifiers that operate on different subsets of features and instances of the stream. This way, ADWIN will reflect potential changes of the data distribution that may not necessarily affect all of the stream but only to subsets of features and labels.

3.1.5 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. Due to the drift detection, both of these ensembles have reset the learning of all base classifiers. However, 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 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.

CHAPTER 4

RESULTS

4.1 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 HESAkNN demonstrate competitive performance compared to state of the art classification methods for multi-label data streams?
- **RQ2:** Is HESAkNN a competitive ensemble in this context when compared strictly to other ensemble algorithms?
- **RQ3:** Is HESAkNN competitive against other cutting edge kNN adaptations for multi-label data streams?
- **RQ4:** How does each of the HESAkNN contributions improve the classification performance?

4.1.1 Experimental setup

4.1.1.1 Algorithms

Table 1 presents a taxonomy of the algorithms used in our experimental study. All algorithms are publicly available in MOA [43]. The source code of HSAkNN is publicly available at `https://github.com/canoalberto/HESAkNN` 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 is 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. On the other hand, kNN methods also have various implementations, as shown in [8]. All algorithms were compared across the same 11 metrics, which will be discussed in further detail in a following section.

4.1.1.2 Datasets

Datasets in our experiments cover a wide range of properties. 30 datasets of up to 269.648k instances, 31.8k features, and 374 labels are evaluated. 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.

Table 1.: Taxonomy of algorithms used in the experiments.

Family	Ref	Acronym	Algorithm
BR + Single	[44]	NB	Naive Bayes
BR + Single	[45]	HT	Hoeffding Tree
BR + Single	[46]	AHT	Adapting Hoeffding Option Tree
BR + Single	[47]	SCD	Single Classifier Drift
BR + Ensemble	[41]	LB	Leveraging Bag
BR + Ensemble	[25]	OB	Oza Bag
BR + Ensemble	[44]	OBA	Oza Bag Adwin
BR + Ensemble	[25]	OBO	Oza Boost
BR + Ensemble	[44]	OBOA	Oza Boost Adwin
BR + Ensemble	[48]	OCB	Online Coordinated Boosting
BR + Ensemble	[49]	DWM	Dynamic Weighted Majority
BR + Ensemble	[50]	AUE	Accuracy Updated Ensemble
BR + Ensemble	[5]	ARF	Adaptive Random Forest
BR + kNN	[51]	kNN	kNN
BR + kNN	[52]	kNNP	kNN PAW
BR + kNN	[52]	kNNPA	kNN PAW ADWIN
BR + kNN	[53]	SAMkNN	Self-Adjusting Memory kNN
AA + Incremental	[54]	BRU	Binary Relevance Updateable
AA + Incremental	[54]	CCU	Classifier Chains Updateable
AA + Incremental	[54]	PSU	Pruned Sets Updateable
AA + Incremental	[54]	RTU	Ranking Threshold Updateable
AA + Incremental	[55]	MLHT	Multilabel Hoeffding Tree
AA + Incremental	[56]	AMR	Adaptive Model Rules
AA + Ensemble	[54]	BML	Bagging ML Updateable
AA + Ensemble	[57]	OBML	Oza Bag ML
AA + Ensemble	[57]	OBAML	Oza Bag Adwin ML
AA + kNN	[11]	MLkNN	ML kNN
AA + kNN	[13]	MLSAMkNN	ML Self-Adjusting Memory kNN
AA + kNN	[14]	MLSAMPkNN	ML Self-Adjusting Memory Punitive kNN
AA + kNN	[8]	MLSAkNN	ML Self-Adjusting kNN
AA + Ensemble + kNN	–	HESAkNN	Homogeneous Self-Adjusting kNN

4.1.1.3 Metrics

Due to the nature of multi-label data streams, traditional evaluation methodologies such as cross-validation are unusable. This gives rise to the use of prequential evaluations to measure model performance. When calculating subset accuracy and Hamming score for HESAkNN, we must account for the total and partial correctness of the multi-label prediction. In order to do so, the following definitions are utilized

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-5k	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-16k	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

over n instances and l labels where the true label set $y = \{y_1, ..., y_l\}$ and predicted label set $\hat{y} = \{\hat{y}_1, ..., \hat{y}_l\}$:

$$Subset\ accuracy = \frac{1}{n}\sum_{i=0}^{n} 1 \mid y_i = \hat{y}_i \tag{4.1}$$

$$Hamming\ score = \frac{1}{n \cdot l}\sum_{i=0}^{n}\sum_{j=0}^{l} 1 \mid y_{ij} = \hat{y}_{ij} \tag{4.2}$$

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 highly imbalanced data distributions as most of the labels are negative, while the challenge is to predict the minority positive instances. Other popular metrics in multi-label classification are also utilized, including: example-based metrics, micro-averaged metrics, and macro-averaged metrics. For each, we can define precision, recall, and F1 [34].

4.1.2 Overall comparison

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 HESAkNN was the top performing model for 9 of the 12 metrics, with significant differences between HESAkNN's performance and the second best algorithm in many of the metrics. For example, subset accuracy is 3 points better (10% improvement) for HESAkNN compared to the second best MLSAkNN. This shows that the ensemble strategy and the original contributions lead to a significant improvement compared to the single base classifier. While HESAkNN's recall values may not be the best, the F1 metric, which balances precision and recall, clearly favors HESAkNN as a well rounded classifier. The following best-performing classifiers are Online Coordinate boosting (OCB) and Adaptive Random Forest (ARF).

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

Algorithm	Su. Acc	H. Sco	Ex. Acc	Ex. Pre	Ex. Rec	Mi. Pre	Mi. Rec	Mi. F1	Ma. Pre	Ma. Rec	Ma. F1	Rank
NB	0.1389	0.8329	0.2298	0.3114	0.4209	0.3893	0.4187	0.3028	0.1980	0.3016	0.1981	20.82
HT	0.2091	0.9136	0.2919	0.5252	0.3301	0.5962	0.3194	0.3696	0.2297	0.1786	0.1788	19.05
AHT	0.2198	0.9157	0.3022	0.5376	0.3393	0.6066	0.3276	0.3790	0.2398	0.1815	0.1862	16.05
SCD	0.2116	0.8746	0.3036	0.4160	0.4418	0.4896	0.4349	0.3743	0.2779	0.2841	0.2473	14.45
LB	0.2717	0.8147	0.3768	0.5300	0.5516	0.4888	0.5404	0.4325	0.2616	0.3655	0.2475	9.82
OB	0.2015	0.8274	0.2894	0.4990	0.4147	0.4654	0.4077	0.3513	0.2187	0.2805	0.1942	19.09
OBA	0.2240	0.8322	0.3123	0.5198	0.4375	0.4798	0.4274	0.3731	0.2449	0.2906	0.2128	15.00
OBO	0.1197	0.7279	0.2503	0.3390	0.5533	0.3181	0.5469	0.3275	0.2514	0.4164	0.2686	16.09
OBOA	0.0258	0.6489	0.1756	0.2108	**0.5961**	0.2016	0.5879	0.2600	0.1970	0.4389	0.2401	20.00
OCB	0.3092	0.9294	0.4150	0.6126	0.4604	0.6341	0.4448	0.4943	0.3355	0.2650	0.2818	5.55
DWM	0.1956	0.8857	0.2914	0.4282	0.4426	0.4878	0.4341	0.3609	0.2314	0.2748	0.2152	16.59
AUE	0.0521	0.5427	0.1580	0.2704	0.5651	0.2250	**0.5713**	0.2092	0.1393	**0.5097**	0.1658	21.82
ARF	0.2978	0.8628	0.4073	0.5760	0.5440	0.5534	0.5286	0.4770	0.3258	0.3416	0.2828	7.45
kNN	0.2376	0.9119	0.3222	0.4955	0.3654	0.5445	0.3530	0.3910	0.2374	0.1815	0.1863	15.77
kNNP	0.2398	0.9142	0.3209	0.5048	0.3606	0.5576	0.3484	0.3901	0.2489	0.1784	0.1869	15.09
kNNPA	0.2590	0.9184	0.3475	0.5287	0.3918	0.5816	0.3767	0.4175	0.2678	0.1934	0.2029	11.61
SAMkNN	0.2627	0.9230	0.3425	0.5936	0.3764	0.6643	0.3594	0.4150	0.3129	0.1834	0.2065	10.36
BRU	0.2120	0.9126	0.3000	0.4934	0.3429	0.5146	0.3306	0.3734	0.1976	0.1843	0.1742	19.55
CCU	0.2266	0.9136	0.3081	0.5060	0.3460	0.5383	0.3323	0.3815	0.2043	0.1850	0.1768	17.05
PSU	0.1605	0.8791	0.2327	0.2995	0.2740	0.3095	0.2549	0.2696	0.1021	0.1344	0.1088	26.59
RTU	0.1645	0.8916	0.1555	0.2789	0.1571	0.4665	0.1507	0.1713	0.1558	0.0984	0.1000	27.45
MLHT	0.1603	0.8791	0.2322	0.2986	0.2734	0.3082	0.2542	0.2687	0.1023	0.1344	0.1088	27.14
AMR	0.1391	0.8329	0.2300	0.3117	0.4209	0.3908	0.4186	0.3029	0.1974	0.3014	0.1975	20.82
BML	0.2207	0.9152	0.3061	0.5371	0.3466	0.5650	0.3341	0.3793	0.2425	0.1801	0.1829	15.82
OBML	0.1599	0.8802	0.2309	0.3055	0.2708	0.3144	0.2514	0.2683	0.0951	0.1272	0.1020	27.55
OBAML	0.2229	0.8925	0.2858	0.3929	0.3243	0.4053	0.3043	0.3263	0.1484	0.1528	0.1364	23.27
MLkNN	0.2303	0.9144	0.3035	0.5466	0.3367	0.6053	0.3239	0.3746	0.2483	0.1670	0.1810	16.27
MLSAMkNN	0.3155	0.9245	0.4042	0.5867	0.4426	0.6198	0.4269	0.4752	0.3183	0.2251	0.2451	7.59
MLSAMPkNN	0.3378	0.9243	0.4354	0.5727	0.4796	0.5869	0.4612	0.5039	0.3163	0.2483	0.2673	6.64
MLSAkNN	0.3515	0.9367	0.4658	0.6223	0.5215	0.6283	0.5040	0.5407	0.3671	0.3126	0.3263	3.53
HESAkNN	**0.3886**	**0.9399**	**0.5038**	**0.6831**	0.5523	**0.7021**	0.5359	**0.5826**	**0.4289**	0.3230	**0.3510**	**2.09**

4.1.3 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. HESAkNN 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 is Online Coordinate boosting (OCB), but there is an 8-points difference with our proposed method. On the other hand, the worst performing ensemble is Oza Boost Adwin.

Next, a detailed statistical analysis comparing the ensemble methods is presented. Figs. 1 and 2 show the Bonferroni-Dunn test for ensemble algorithms on the subset

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

Dataset	LB	OB	OBA	OBO	OBOA	OCB	DWM	AUE	ARF	BML	OBML	OBAML	HESAkNN
Birds	0.4177	0.4404	0.4404	0.1920	0.1848	0.4207	0.3642	0.0706	**0.4580**	0.4250	0.4510	0.4510	0.4544
Virus	0.3041	0.1456	0.1456	0.2543	0.1279	0.3947	0.2168	0.0000	0.3146	0.3498	0.2592	0.2592	**0.6706**
Flags	0.1030	0.1148	0.1148	0.0797	0.0797	0.1137	0.1042	0.0092	0.1179	0.1054	**0.1294**	**0.1294**	0.1228
Scene	0.6621	0.3488	0.5149	0.5170	0.0450	0.8368	0.4968	0.0615	0.7948	0.3574	0.3426	0.4931	**0.8745**
Enron	0.0318	0.0355	0.0305	0.0028	0.0011	0.0666	0.0338	0.0007	0.0547	0.0825	0.0884	**0.1089**	0.0949
Genbase	0.7435	0.3652	0.3652	0.0156	0.0156	0.7651	0.2337	0.1207	0.4515	0.3698	0.2621	0.2382	**0.9178**
Medical	0.4754	0.2760	0.2760	0.0000	0.0000	0.2619	0.0014	0.1973	**0.5028**	0.2851	0.1479	0.1479	0.3534
Water-qual	0.0172	0.0047	0.0056	0.0021	0.0019	0.0099	0.0021	0.0005	0.0194	0.0040	0.0072	0.0120	**0.0265**
Corel-5k	0.0208	0.0069	0.0075	0.0000	0.0000	0.0315	0.0000	0.0002	0.0400	0.0092	0.0089	0.0230	**0.0545**
Eukaryote	0.6257	0.4059	0.5052	0.3447	0.0000	0.7031	0.4314	0.1954	0.6860	0.4060	0.1616	0.7522	**0.8134**
Plant	0.3168	0.2285	0.2477	0.1192	0.0000	0.7255	0.2377	0.0278	0.4295	0.2358	0.3135	0.5386	**0.7619**
Reuters	0.0795	0.0650	0.0650	0.0000	0.0000	0.0518	0.0121	0.0360	0.1466	0.0812	0.2109	0.1909	**0.2400**
Mediamill	0.1174	0.0689	0.0629	0.0293	0.0000	0.0952	0.0029	0.0545	0.1458	0.0709	0.0532	0.0826	**0.1895**
Ohsumed	0.1383	0.1523	0.1524	0.1230	0.0164	0.1315	0.1266	0.0423	0.1787	0.1239	0.1617	0.1574	**0.6798**
CAL-500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Yelp	0.5328	0.3401	0.4594	0.4610	0.0218	0.4463	0.4020	0.1663	0.6313	0.3467	0.2375	0.4132	**0.6651**
Slashdot	0.0262	0.1019	0.1019	0.0000	0.0000	0.0612	0.0290	0.0580	0.0962	0.0107	0.1298	0.1372	**0.1434**
Human	0.5099	0.1970	0.2998	0.3761	0.0378	0.6363	0.3635	0.0536	0.5356	0.1981	0.2068	0.6480	**0.7513**
Langlog	0.0253	0.0273	0.0273	0.0000	0.0000	0.1462	0.1538	0.0070	0.1464	0.1388	0.1381	0.1388	**0.1779**
Gnegative	0.6298	0.5808	0.5965	0.2424	0.0000	0.8869	0.6054	0.1797	0.7354	0.6016	0.4159	0.4159	**0.8956**
CHD	0.1838	0.1562	0.1562	0.1288	0.1243	0.1452	0.1055	0.0058	0.1714	0.1554	0.1430	0.1430	**0.2293**
Stackex	0.0069	0.0156	0.0156	0.0000	0.0000	0.0196	0.0124	0.0062	0.0037	0.0024	**0.0276**	0.0111	0.0144
Corel-16k	0.0577	0.0130	0.0144	0.0341	0.0000	0.0869	0.0008	0.0005	0.1011	0.0220	0.0206	0.0748	**0.1215**
Imdb	0.0121	0.0031	0.0067	0.0129	0.0032	0.0302	0.0122	0.0002	0.0291	0.0213	0.1081	**0.1101**	0.0934
Nuswide-C	0.2632	0.2397	0.2503	0.1978	0.1003	0.2129	0.1764	0.2211	**0.2687**	0.2399	0.1330	0.2487	0.2401
Nuswide-B	0.2263	0.2256	0.2251	0.1972	0.0130	0.2109	0.2265	0.0036	**0.2536**	0.2196	0.1184	0.2483	0.2394
Yahoo-soc	0.0036	0.0204	0.0061	0.0000	0.0000	0.1016	0.0271	0.0021	0.0022	0.1441	**0.2818**	**0.2818**	0.1855
Eurlex-sm	0.0000	0.0000	0.0000	0.0000	0.0000	0.0726	0.0006	0.0000	0.0000	**0.1576**	0.1411	0.1352	0.1181
Hypersphe	0.6218	0.4724	**0.6281**	0.1818	0.0000	0.6109	0.4934	0.0135	0.6215	0.4636	0.0287	0.0287	0.5289
Hypercube	0.9973	0.9937	0.9976	0.0795	0.0000	0.9991	0.9954	0.0298	0.9979	0.9929	0.0675	0.0675	**0.9993**
Average	0.2717	0.2015	0.2240	0.1197	0.0258	0.3092	0.1956	0.0521	0.2978	0.2207	0.1599	0.2229	**0.3886**
Rank	5.6774	7.6774	6.5484	9.9839	12.2258	4.7742	8.2742	11.1935	3.8387	6.8871	6.8548	5.0000	**2.0645**

accuracy and Hamming score respectively (multiple-algorithm comparison test). The figures illustrate the rank of the algorithms and the critical distance for $\alpha = 0.01$. Algorithms outside the interval of the critical distance are said to perform statistically worse than the control method. According to the multiple-comparison test, it cannot be said that there are statistically significant differences when comparing HESAkNN with Adaptive Random Forest (ARF) and Online Coordinated Boosting (OCB) for both metrics. However, statistically significant differences exist for the rest of the ensembles.

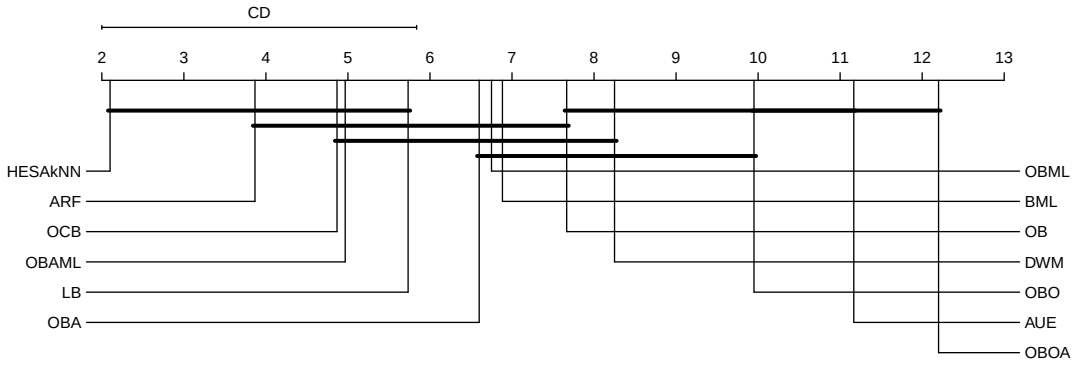


Fig. 1.: Bonferroni-Dunn for subset accuracy on ensembles.

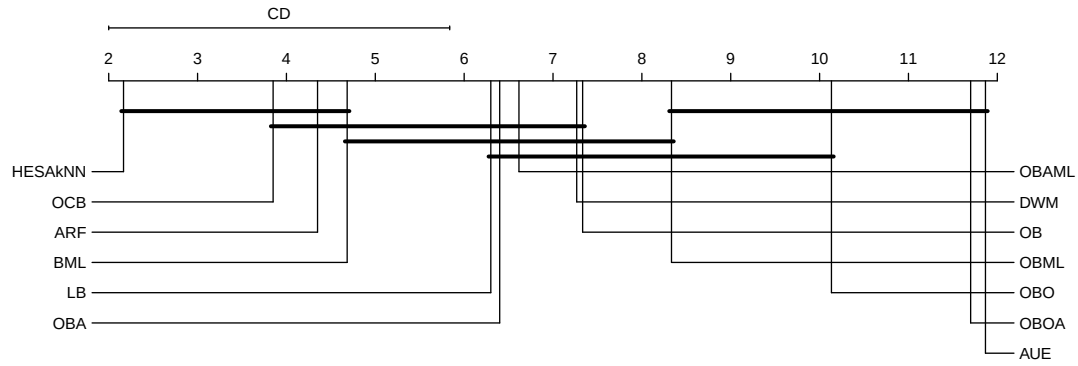
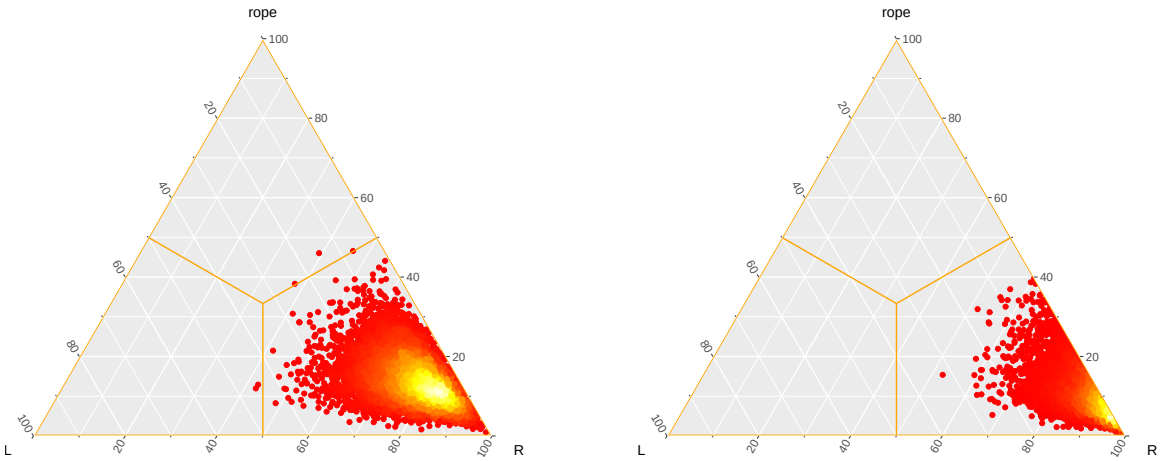


Fig. 2.: Bonferroni-Dunn for Hamming score on ensembles.

Table 5 shows the p-values reported for the Wilcoxon signed-rank test on the main performance metrics: subset accuracy, Hamming score, micro-F1, and macro-F1, where p-values < 0.01 indicate statistically significant differences between HESAkNN 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 and the Adaptive Random Forest model. Finally, Fig. 3 illustrates the Bayesian sign test. 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 bet-

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

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



(a) ARF vs HESAkNN. (b) OCB vs HESAkNN.

Fig. 3.: Bayesian sign test: subset accuracy on top ensembles.

ter performance for HESAkNN and the remaining side for the opposing algorithm. Based on the results shown in the figure, HESAkNN outperforms Adaptive Random Forest at almost every evaluation and Online Coordinated Boosting every time.

Fig. 4 helps to visualize why HESAkNN is rating well across these various tests and metrics. It is shown in this figure how subset accuracy varies with instance arrival over time, adapting to concept drift and changing properties of the stream.

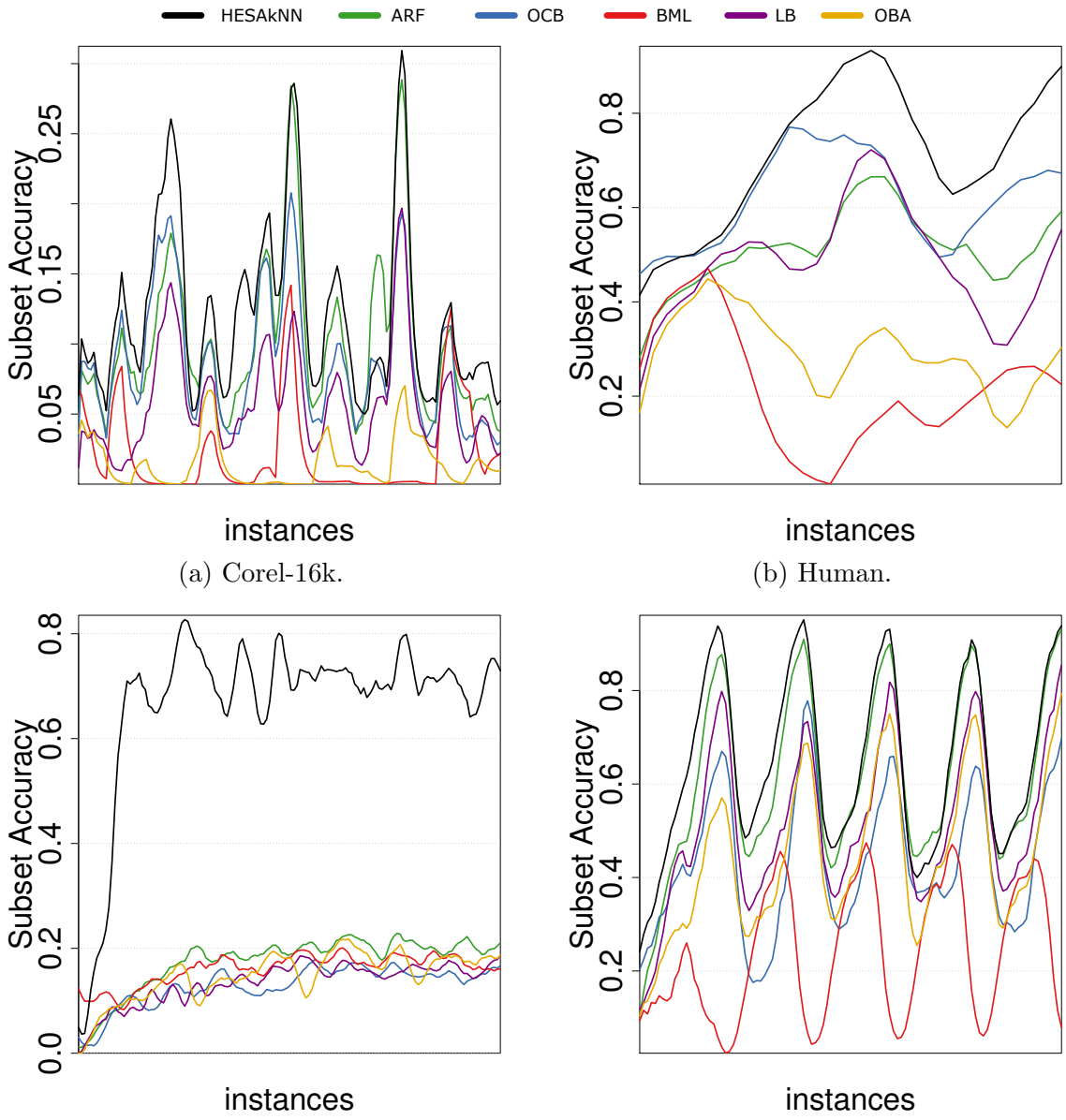
Notice how HESAkNN is the first to adapt and maintains the highest subset accuracy for the most prolonged duration. This can contribute to an explanation as to why there are performance gains with HESAkNN. First, adaption happens early. As soon as drifts are detected, changes are made, and metrics for meta evaluations begin to be gathered. Second, the ensemble is diverse and contains competent classifiers for longer amounts of time due to frequent evaluations and updates. In short, HESAkNN is more adaptable to changes in concept and varying difficulties in the data.

4.1.4 Nearest neighbors comparison

The third experimental study aims at providing an in-depth evaluation and comparison of classifiers based on the nearest neighbor to address RQ3. As shown in Table 6, HESAkNN outperforms the kNN family of algorithms on 17 out of the 30 provided datasets. It is also shown to have the most competitive average accuracy and the best rank. Showing that at a minimum HESAkNN is a top running contender not only for ensemble algorithms but also adapted kNN algorithms. HESAkNN shows significantly better performance than MLSAkNN, demonstrating the advantages of the ensemble approach and its methodologies for adaptation to concept drift.

Fig. 5 shows that the Bonferroni-Dunn test cannot find significant statistical differences exist between HESAkNN, MLSAkNN, MLSAMPkNN, and MLSAMkNN based on subset accuracy. Fig. 6 shows the Bonferroni-Dunn test utilizing hamming distance, it reveals no statistical differences between HESAkNN, MLSAkNN, SAMkNN, MLkNN, and MLSAMkNN. After considering both tests, it can be seen that our combined metrics show no overall statistical difference between HESAkNN, MLSAkNN, and MLSAMkNN, which means that a significant statistical difference exists for the remaining kNN methods.

Table 7 reveals something more interesting in the kNN family compared to the



(a) Corel-16k.

(b) Human.

(c) Ohsumed.

(d) Yelp.

Fig. 4.: Comparison of top performing ensembles: subset accuracy on four datasets.

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 component classifier of HESAkNN, it is understandable that these values would be the most similar. It serves as an

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

Dataset	kNN	kNNP	kNNPA	SAMkNN	MLkNN	MLSAMkNN	MLSAMPkNN	MLSAkNN	HESAkNN
Birds	0.4792	**0.4814**	**0.4814**	0.4676	0.4729	0.4747	0.4683	0.4542	0.4544
Virus	0.4827	0.4684	0.4684	0.4767	0.5334	0.5530	0.5659	0.6324	**0.6706**
Flags	0.1165	0.1088	0.1088	0.0409	0.0869	0.0753	0.0681	**0.1299**	0.1228
Scene	0.5443	0.6492	0.6484	0.6792	0.5287	0.8279	0.8461	0.8458	**0.8745**
Enron	0.0891	0.0953	0.0861	0.0870	0.0724	0.0976	**0.1005**	0.0952	0.0949
Genbase	0.7617	0.7498	0.7498	0.2091	0.7543	0.8832	0.8785	0.9118	**0.9178**
Medical	0.3284	0.3151	0.3151	0.2429	0.3701	0.3811	**0.4180**	0.3663	0.3534
Water-qual	0.0126	0.0177	0.0171	0.0199	0.0125	0.0159	0.0207	**0.0572**	0.0265
Corel-5k	0.0035	0.0005	0.0049	0.0043	0.0048	0.0216	0.0346	0.0465	**0.0545**
Eukaryote	0.3429	0.2836	0.4480	0.6255	0.2988	0.6982	0.7170	0.7386	**0.8134**
Plant	0.1469	0.1545	0.1606	0.3321	0.1915	0.4170	0.5657	0.6164	**0.7619**
Reuters	0.1803	0.1889	0.1889	0.1365	0.1661	0.1907	**0.2495**	0.2392	0.2400
Mediamill	0.1019	0.1073	0.1055	0.1458	0.0873	0.1452	0.1603	0.1517	**0.1895**
Ohsumed	0.0066	0.0051	0.0051	0.0113	0.0139	0.0265	0.0315	0.0326	**0.6798**
CAL-500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Yelp	0.1988	0.2262	0.4069	0.5174	0.1528	0.5028	0.5674	0.6249	**0.6651**
Slashdot	0.1400	0.1393	0.1390	0.0661	0.0694	0.1887	**0.2047**	0.1891	0.1434
Human	0.1973	0.1968	0.3281	0.4981	0.1186	0.5785	0.6447	0.6683	**0.7513**
Langlog	0.1791	0.1787	0.1787	0.1476	0.1472	0.1590	**0.1877**	0.1756	0.1779
Gnegative	0.5212	0.5343	0.5918	0.7255	0.5790	0.7538	0.8226	0.8419	**0.8956**
CHD	0.1478	0.1657	0.1657	0.1202	0.1123	0.1534	0.1398	**0.2307**	0.2293
Stackex	0.0109	0.0068	0.0068	0.0134	0.0075	0.0100	0.0139	0.0128	**0.0144**
Corel-16k	0.0108	0.0064	0.0195	0.0241	0.0092	0.0699	0.0867	0.1167	**0.1215**
Imdb	0.0339	0.0311	0.0309	0.0171	0.0032	0.0447	0.0661	0.0717	**0.0934**
Nuswide-C	0.2291	0.2360	0.2378	**0.2662**	0.2317	0.2555	0.2466	0.2509	0.2401
Nuswide-B	0.1573	0.1618	0.1565	0.2594	0.2221	**0.2600**	0.2424	0.2486	0.2394
Yahoo-soc	0.1214	0.1262	0.1124	0.1547	0.0951	0.1177	0.1476	0.1430	**0.1855**
Eurlex-sm	0.0699	0.0874	0.0834	0.0722	0.0569	0.0428	0.1300	**0.1335**	0.1181
Hypersphe	0.5184	0.4799	0.5272	0.5239	0.5129	0.5216	0.5105	0.5231	**0.5289**
Hypercube	0.9950	0.9919	0.9963	0.9979	0.9982	0.9985	0.9969	0.9960	**0.9993**
Average	0.2376	0.2398	0.2590	0.2627	0.2303	0.3155	0.3378	0.3515	**0.3886**
Rank	6.4667	6.5000	6.1333	5.7000	7.1667	4.3667	3.3333	3.0000	**2.3333**

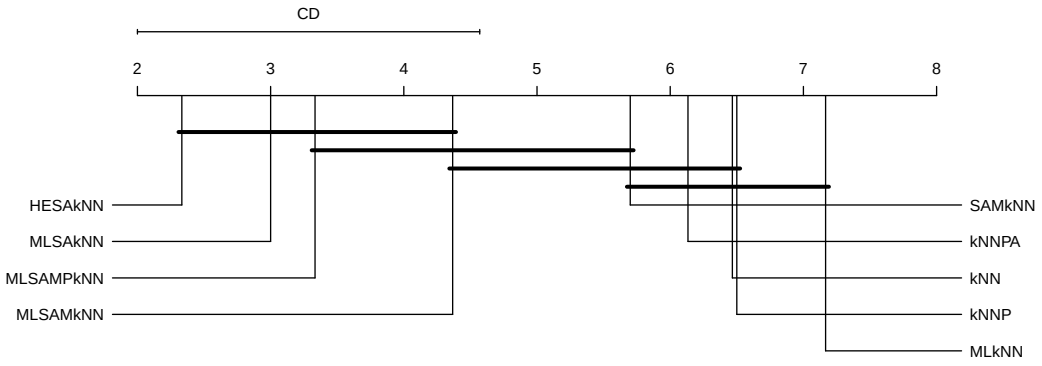


Fig. 5.: Bonferroni-Dunn for subset accuracy on nearest neighbors.

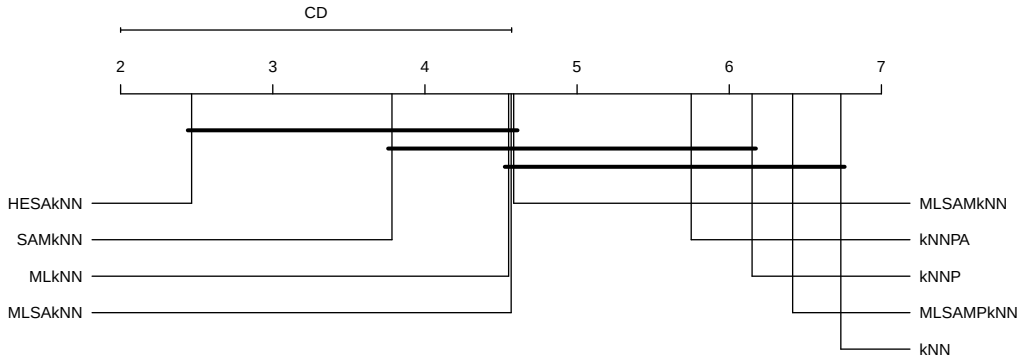


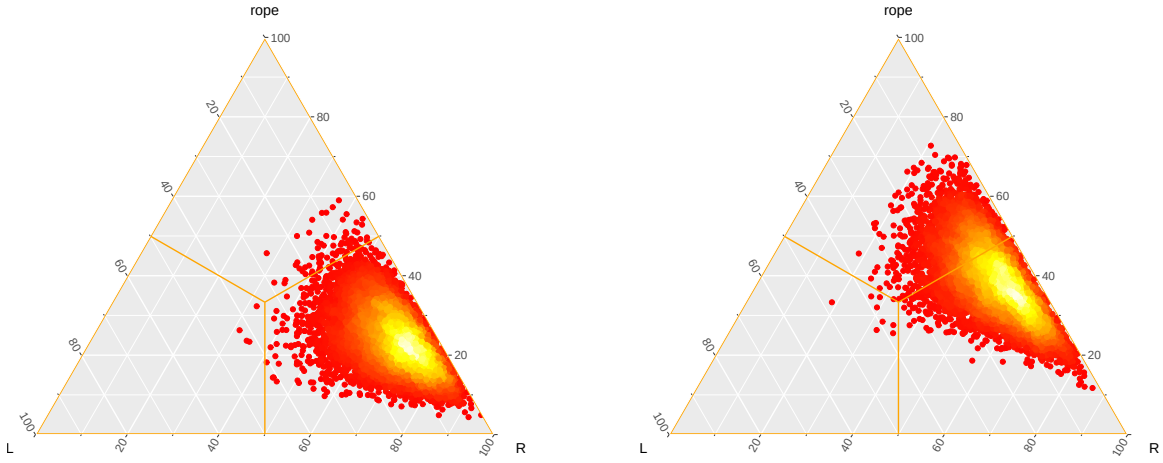
Fig. 6.: Bonferroni-Dunn for Hamming score on nearest neighbors.

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

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

important reflection to note that the p-values for MLSAkNN should help establish the statistical difference of HESAkNN based on only the ensemble mechanisms.

Fig. 7 shows the Bayesian analysis and compares probabilities of outperformance between HESAkNN and MLSAkNN as well as HESAkNN and MLSAMPkNN. The big takeaway from this test is to reveal that HESAkNN 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 HESAkNN. 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



(a) MLSAMPkNN vs HESAkNN. (b) MLSAkNN vs HESAkNN.

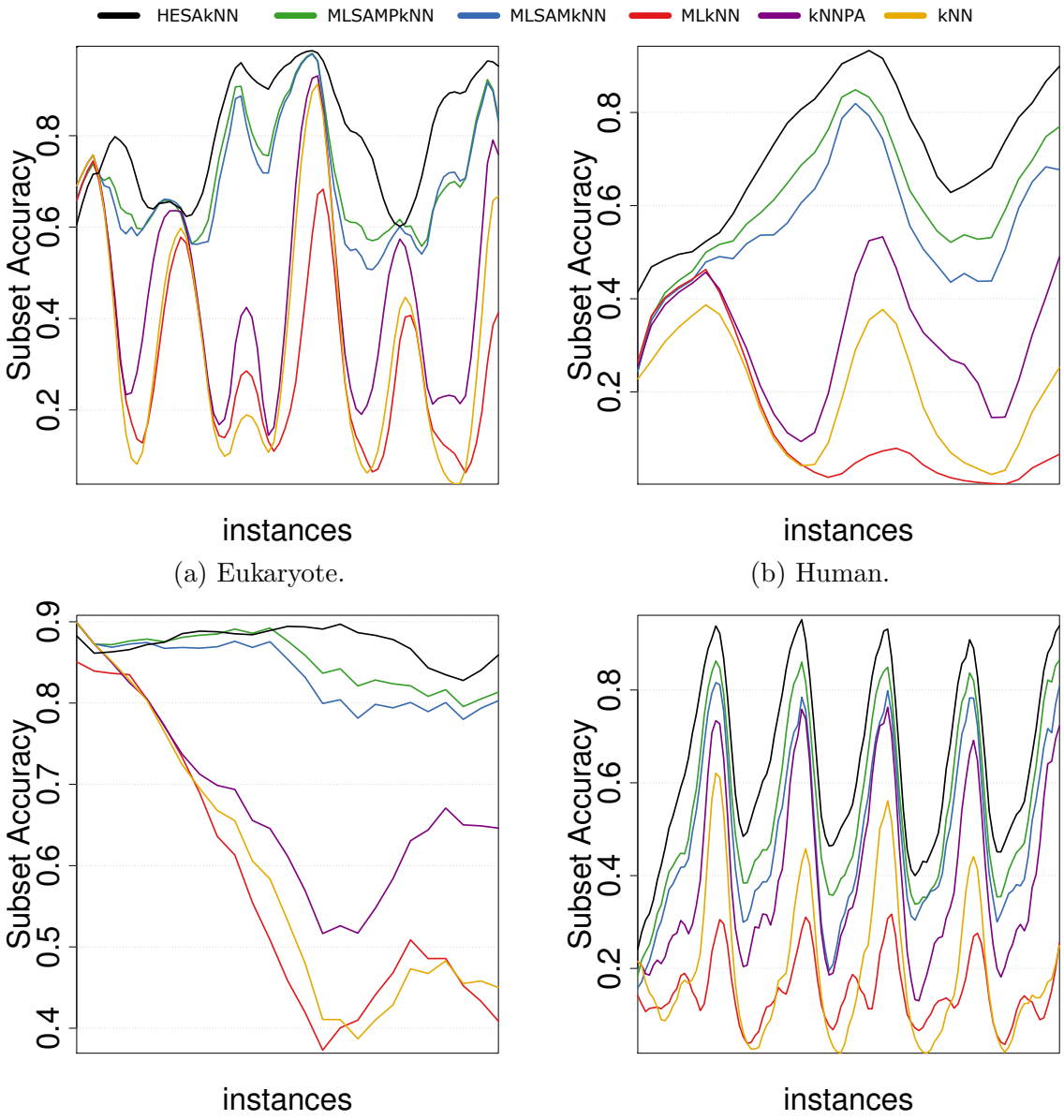
Fig. 7.: Bayesian sign test: subset accuracy on top nearest neighbors.

minimal.

The visualizations in Fig. 8 follow similar trends as we have seen with the ensemble algorithms. While in this batch of graphs HESAkNN is not always initially the clear best predictor, it still maintains the best performance over the most prolonged interval of instances. These graphs demonstrate that HESAkNN might improve through learning mechanisms focused on stream initialization when the least amount of information is known. However, general performance throughout the data stream still verifies the ability of HESAkNN to better adapt to drifting, complex data.

4.1.5 Contributions of HESAkNN

The fourth and last experiment aims at evaluating how each of the main four contributions of HESAkNN help to improve the classification and by what margin. Fig. 9 shows the prequential subset accuracy over time for four example datasets. It illustrates the performance of HESAkNN and four variants of the algorithm without each of these contributions: no background ensemble, no feature subspaces, no online bagging, and a single classifier. The performance for the Eukaryote and Human

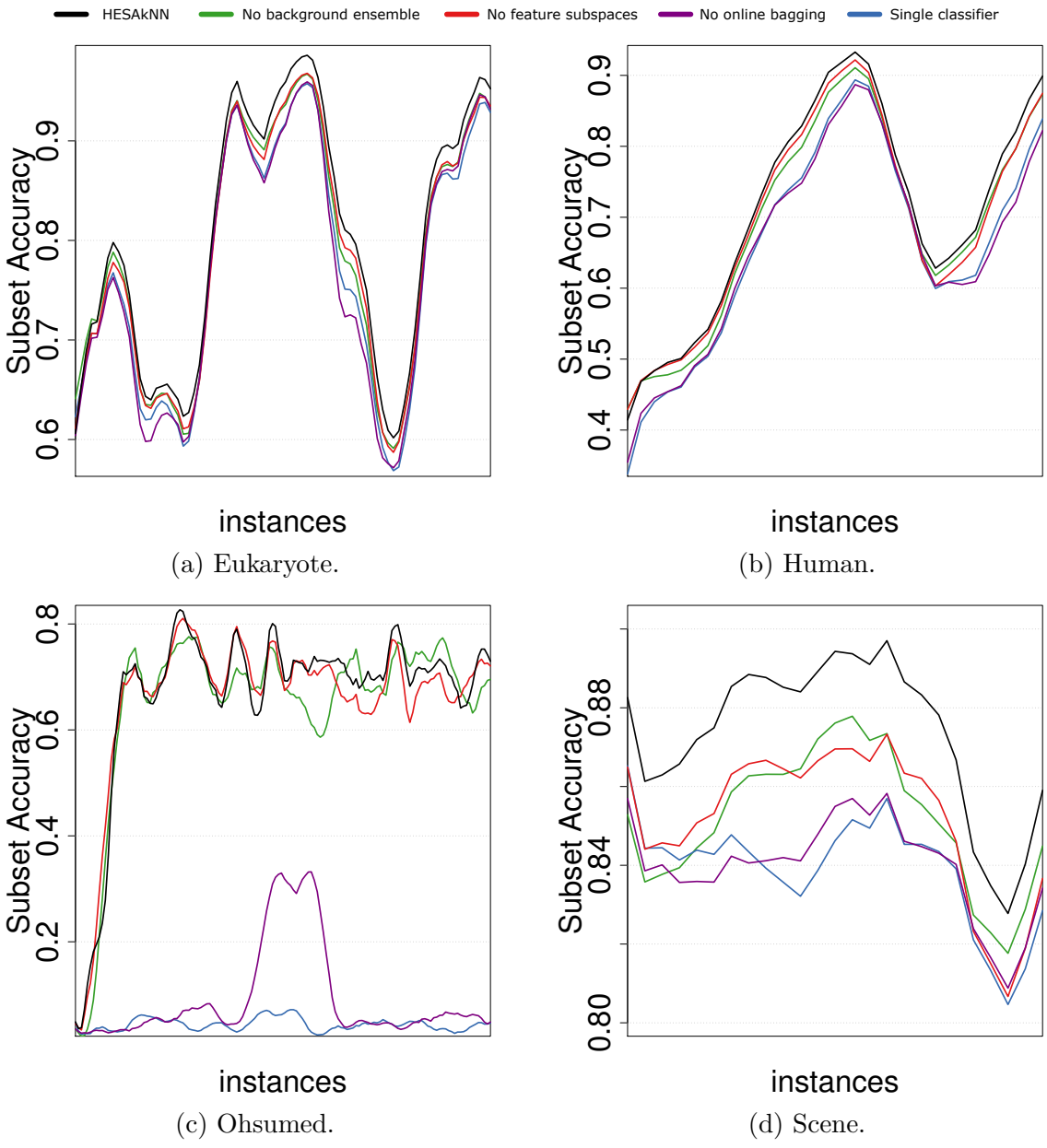


(a) Eukaryote. (b) Human.

(c) Scene. (d) Yelp.

Fig. 8.: Comparison of nearest neighbor approaches: subset accuracy on four datasets.

datasets are very similar for all the variants. Still, HESAkNN shows the best overall subset accuracy, 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 differ-



(a) Eukaryote. (b) Human.

(c) Ohsumed. (d) Scene.

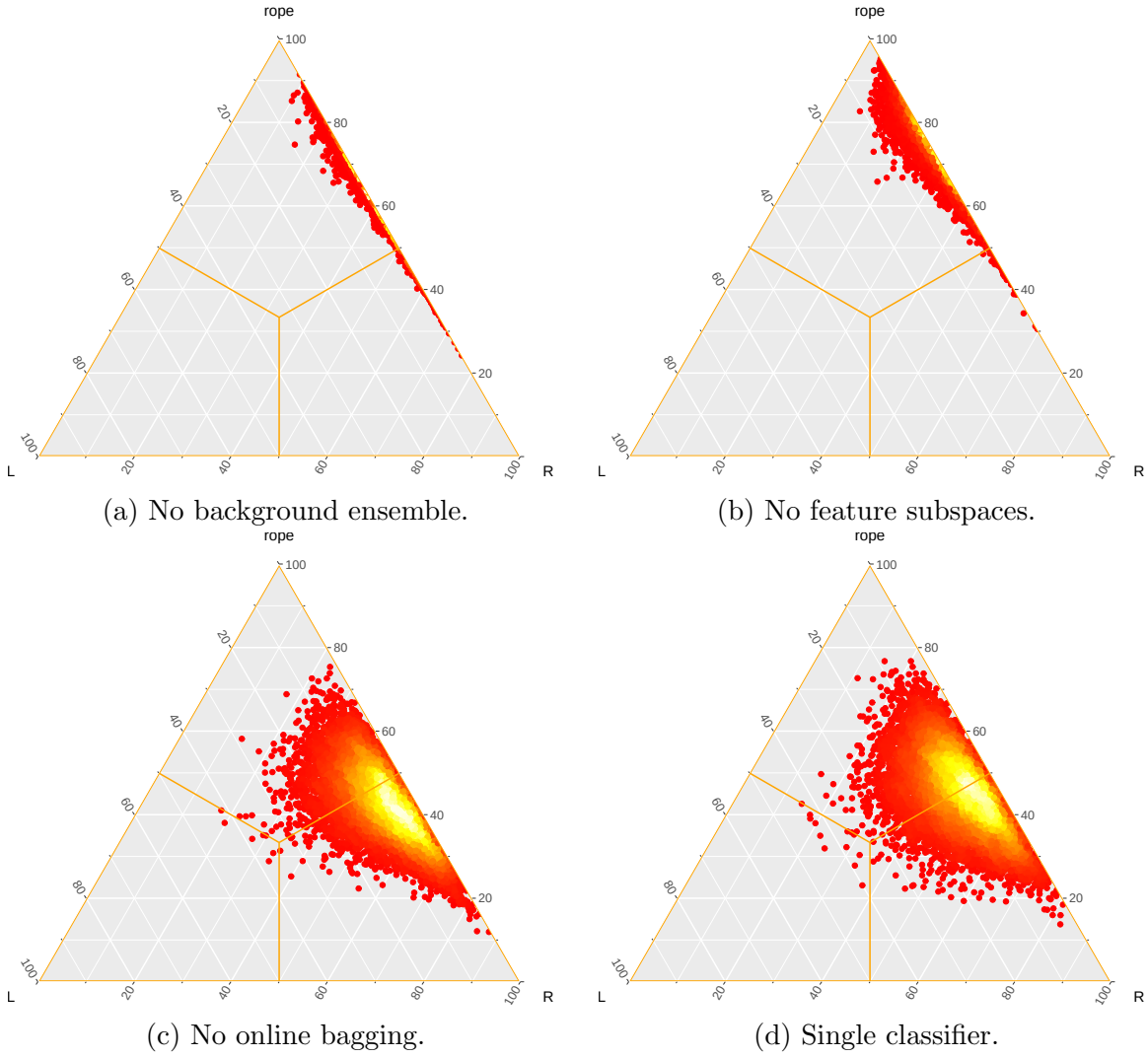
Fig. 9.: Comparison of HESAkNN four main contributions on four datasets.

ence 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 HESAkNN a robust ensemble.

Fig. 10 show the results of the Bayesian sign test of HESAkNN vs each of the four variants without a key component. 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 HESAkNN (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.

4.2 Conclusions and future work

This thesis introduced HESAkNN, a homogeneous ensemble of self-adjusting kNNs for multi-label data stream classification. HESAkNN combines learners who are naturally adaptive to concept drift. Includes modules for both instance and feature subspaces learning to improve the ensemble diversity. It also contains a collection of ADWIN detectors for explicit drift detection and a background ensemble mechanism to help combat and adapt to concept drift. These combined mechanisms allow HESAkNN to be highly adaptive to concept drift for each individual label and overcome other various data difficulties. A thorough experimental study has shown how competitive HESAkNN is against 30 different models across 30 datasets and 12 multi-label metrics. Achieving top performance for 9 out of 12 metrics. A closer look at how HESAkNN compares to other ensemble algorithms confirmed through various statistical analyses and empirical metrics that HESAkNN is a front runner for this context. Finally another confirmation that the ensemble performance of MLSAkNNs outperforms other various modified kNN algorithms, demonstrating the predictive power of ensemble techniques.



(a) No background ensemble.

(b) No feature subspaces.

(c) No online bagging.

(d) Single classifier.

Fig. 10.: Bayesian sign test: subset accuracy on HESAkNN four main contributions.

As future work, there are still more adaptive mechanisms that could potentially benefit HESAkNN. This could include adaptive windowing and additional hyperparameter adjustment, or dynamic ensemble sizing to increase the amount of knowledge gained during meta evaluations.

Appendix A

ABBREVIATIONS

VCU	Virginia Commonwealth University
RVA	Richmond Virginia
NB	Naive Bayes
HT	Hoeffding Tree
AHT	Adapting Hoeffding Option Tree
SCD	Single Classifier Drift
LB	Leveraging Bag
OB	Oza Bag
OBA	Oza Bag Adwin
OBO	Oza Boost
OBOA	Oza Boost Adwin
OCB	Online Coordinated Boosting
DWM	Dynamic Weighted Majority
AUE	Accuracy Updated Ensemble
ARF	Adaptive Random Forest
kNN	kNN
kNNP	kNN PAW
kNNPA	kNN PAW ADWIN
SAMkNN	Self-Adjusting Memory kNN
BRU	Binary Relevance Updateable
CCU	Classifier Chains Updateable
PSU	Pruned Sets Updateable
RTU	Ranking Threshold Updateable
MLHT	Multilabel Hoeffding Tree
AMR	Adaptive Model Rules
BML	Bagging ML Updateable
OBML	Oza Bag ML
OBAML	Oza Bag Adwin ML
MLkNN	ML kNN
MLSAMkNN	ML Self-Adjusting Memory kNN
MLSAMPkNN	ML Self-Adjusting Memory Punitive kNN
MLSAkNN	ML Self-Adjusting kNN
HESAkNN	Homogeneous Self-Adjusting kNN

REFERENCES

[1] Min-Ling Zhang and Zhi-Hua Zhou. "A review on multi-label learning algorithms". In: *IEEE Transactions on Knowledge and Data Engineering* 26.8 (2013), pp. 1819–1837.

[2] Jian Wu et al. "Multi-label active learning for image classification". In: *IEEE International Conference on Image Processing*. 2014, pp. 5227–5231.

[3] Eftychios Protopapadakis, Iason Katsamenis, and Anastasios Doulamis. "Multi-label deep learning models for continuous monitoring of road infrastructures". In: *ACM International Conference on PErvasive Technologies Related to Assistive Environments*. 2020, pp. 1–7.

[4] Zhi-Hao Wu et al. "Balanced multi-label propagation for overlapping community detection in social networks". In: *Journal of Computer Science and Technology* 27.3 (2012), pp. 468–479.

[5] Heitor M Gomes et al. "Adaptive random forests for evolving data stream classification". In: *Machine Learning* 106.9 (2017), pp. 1469–1495.

[6] Isah A Lawal and Salihu A Abdulkarim. "Adaptive SVM for data stream classification". In: *South African Computer Journal* 29.1 (2017), pp. 27–42.

[7] A. Cano and B. Krawczyk. "Kappa Updated Ensemble for Drifting Data Stream Mining". In: *Machine Learning* 109.1 (2020), pp. 175–218.

[8] Martha Roseberry et al. "Self-adjusting k nearest neighbors for continual learning from multi-label drifting data streams". In: *Neurocomputing* 442 (2021), pp. 10–25.

[9] Min-Ling Zhang et al. “Binary relevance for multi-label learning: an overview”. In: *Frontiers of Computer Science* 12.2 (2018), pp. 191–202.

[10] Jinghua Liu et al. “Feature selection for multi-label learning with streaming label”. In: *Neurocomputing* 387 (2020), pp. 268–278.

[11] Min-Ling Zhang and Zhi-Hua Zhou. “ML-KNN: A lazy learning approach to multi-label learning”. In: *Pattern recognition* 40.7 (2007), pp. 2038–2048.

[12] Koby Crammer and Yoram Singer. “A family of additive online algorithms for category ranking”. In: *Journal of Machine Learning Research* 3 (2003), pp. 1025–1058.

[13] Martha Roseberry and Alberto 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.

[14] Martha Roseberry, Bartosz Krawczyk, and Alberto Cano. “Multi-label punitive kNN with self-adjusting memory for drifting data streams”. In: *ACM Transactions on Knowledge Discovery from Data* 13.6 (2019), pp. 1–31.

[15] Joao Gama and Petr Kosina. “Recurrent concepts in data streams classification”. In: *Knowledge and Information Systems* 40.3 (2014), pp. 489–507.

[16] Alberto Fernández et al. “SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary”. In: *Journal of Artificial Intelligence Research* 61 (2018), pp. 863–905.

[17] Chris Drummond, Robert 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.

[18] Addisson Salazar, Gonzalo Safont, and Luis Vergara. “Semi-supervised learning for imbalanced classification of credit card transaction”. In: *International Joint Conference on Neural Networks*. 2018, pp. 1–7.

[19] Min-Ling Zhang et al. “Towards class-imbalance aware multi-label learning”. In: *IEEE Transactions on Cybernetics* (2020).

[20] Xiulin Zheng et al. “A survey on multi-label data stream classification”. In: *IEEE Access* 8 (2019), pp. 1249–1275.

[21] Hang Zhang et al. “Resample-based ensemble framework for drifting imbalanced data streams”. In: *IEEE Access* 7 (2019), pp. 65103–65115.

[22] Min-Ling Zhang and Lei Wu. “Lift: Multi-label learning with label-specific features”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.1 (2014), pp. 107–120.

[23] Wenyu Zang et al. “Comparative study between incremental and ensemble learning on data streams: Case study”. In: *Journal Of Big Data* 1.1 (2014), pp. 1–16.

[24] Bartosz Krawczyk et al. “Ensemble learning for data stream analysis: A survey”. In: *Information Fusion* 37 (2017), pp. 132–156.

[25] Nikunj C Oza and Stuart J Russell. “Online bagging and boosting”. In: *International Workshop on Artificial Intelligence and Statistics*. 2001, pp. 229–236.

[26] Yu Sun et al. “Online ensemble learning of data streams with gradually evolved classes”. In: *IEEE Transactions on Knowledge and Data Engineering* 28.6 (2016), pp. 1532–1545.

[27] Tinofirei Museba, Fulufhelo Nelwamondo, and Khmaies Ouahada. "An Adaptive Heterogeneous Online Learning Ensemble Classifier for Nonstationary Environments". In: *Computational Intelligence and Neuroscience* 2021 (2021).

[28] B. Krawczyk and A. Cano. "Online Ensemble Learning with Abstaining Classifiers for Drifting and Noisy Data Streams". In: *Applied Soft Computing* 68 (2018), pp. 677–692.

[29] Elena Montanes et al. "Dependent binary relevance models for multi-label classification". In: *Pattern Recognition* 47.3 (2014), pp. 1494–1508.

[30] Adriano Rivolli et al. "An empirical analysis of binary transformation strategies and base algorithms for multi-label learning". In: *Machine Learning* 109.8 (2020), pp. 1509–1563.

[31] Jesse Read et al. "Classifier chains for multi-label classification". In: *Machine learning* 85.3 (2011), p. 333.

[32] Lior Rokach, Alon Schclar, and Ehud Itach. "Ensemble methods for multi-label classification". In: *Expert Systems with Applications* 41.16 (2014), pp. 7507–7523.

[33] Vu-Linh Nguyen et al. "On Aggregation in Ensembles of Multilabel Classifiers". In: *International Conference on Discovery Science*. 2020, pp. 533–547.

[34] Jose M Moyano et al. "Review of ensembles of multi-label classifiers: models, experimental study and prospects". In: *Information Fusion* 44 (2018), pp. 33–45.

[35] Qingyao Wu et al. "ML-FOREST: A multi-label tree ensemble method for multi-label classification". In: *IEEE transactions on knowledge and data engineering* 28.10 (2016), pp. 2665–2680.

[36] Jun Huang et al. “Improving multi-label classification with missing labels by learning label-specific features”. In: *Information Sciences* 492 (2019), pp. 124–146.

[37] Chunyu Zhang and Zhanshan Li. “Multi-label learning with label-specific features via weighting and label entropy guided clustering ensemble”. In: *Neurocomputing* 419 (2021), pp. 59–69.

[38] Yange Sun, Han Shao, and Shasha Wang. “Efficient ensemble classification for multi-label data streams with concept drift”. In: *Information* 10.5 (2019), p. 158.

[39] Ricardo Sousa and João Gama. “Multi-label classification from high-speed data streams with adaptive model rules and random rules”. In: *Progress in Artificial Intelligence* 7.3 (2018), pp. 177–187.

[40] Qinghua Wu et al. “MapReduce-based adaptive random forest algorithm for multi-label classification”. In: *Neural Computing and Applications* 31.12 (2019), pp. 8239–8252.

[41] Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. “Leveraging bagging for evolving data streams”. In: *European Conference on Machine Learning and Knowledge Discovery in Databases*. 2010, pp. 135–150.

[42] Albert Bifet and Ricard Gavalda. “Learning from time-changing data with adaptive windowing”. In: *SIAM international conference on data mining*. 2007, pp. 443–448.

[43] Albert Bifet et al. “MOA: Massive Online Analysis”. In: *Journal of Machine Learning Research* 11 (2010), pp. 1601–1604.

[44] Albert Bifet et al. “New ensemble methods for evolving data streams”. In: *ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 139–148.

[45] Pedro Domingos and Geoff Hulten. “Mining high-speed data streams”. In: *ACM SIGKDD international conference on Knowledge discovery and data mining*. 2000, pp. 71–80.

[46] Bernhard Pfahringer, Geoffrey Holmes, and Richard Kirkby. “New options for hoeffding trees”. In: *Australasian Joint Conference on Artificial Intelligence*. 2007, pp. 90–99.

[47] Manuel Baena-Garcıa et al. “Early drift detection method”. In: *International Workshop on Knowledge Discovery from Data Streams*. Vol. 6. 2006, pp. 77–86.

[48] Raphael Pelossof et al. “Online coordinate boosting”. In: *International Conference on Computer Vision Workshops*. 2009, pp. 1354–1361.

[49] J Zico Kolter and Marcus A Maloof. “Dynamic weighted majority: An ensemble method for drifting concepts”. In: *The Journal of Machine Learning Research* 8 (2007), pp. 2755–2790.

[50] Dariusz Brzezinski and Jerzy Stefanowski. “Reacting to different types of concept drift: The accuracy updated ensemble algorithm”. In: *IEEE Transactions on Neural Networks and Learning Systems* 25.1 (2013), pp. 81–94.

[51] Philipp Kranen et al. “Stream data mining using the MOA framework”. In: *International Conference on Database Systems for Advanced Applications*. 2012, pp. 309–313.

[52] Albert Bifet et al. “Efficient data stream classification via probabilistic adaptive windows”. In: *ACM symposium on applied computing*. 2013, pp. 801–806.

[53] V. Losing, B. Hammer, and H. Wersing. "KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift". In: *IEEE International Conference on Data Mining*. 2016, pp. 291–300.

[54] Jesse Read et al. "MEKA: A Multi-label/Multi-target Extension to Weka". In: *Journal of Machine Learning Research* 17.21 (2016), pp. 1–5.

[55] Jesse Read et al. "Streaming multi-label classification". In: *Workshop on Applications of Pattern Analysis*. 2011, pp. 19–25.

[56] Ricardo Sousa and João Gama. "Online Multi-label Classification with Adaptive Model Rules". In: *Conference of the Spanish Association for Artificial Intelligence*. 2016, pp. 58–67.

[57] Jesse Read et al. "Scalable and efficient multi-label classification for evolving data streams". In: *Machine Learning* 88.1-2 (2012), pp. 243–272.

VITA

Gavin Joseph Alberghini was born February 17, 1999, in Pensacola Florida and is an American citizen. He graduated from Hickory High School, Chesapeake, Virginia in 2017. He received his Bachelor of Science in Computer Science from Virginia Commonwealth University, Richmond Virginia in 2020. This thesis has been submitted to the 2021 Neurocomputing journal.