# Multi-Label Transfer Learning in Non-Stationary Data Streams

Honghui Du*, Leandro L. Minku†, Aonghus Lawlor*, Huiyu Zhou‡

*Insight Centre for Data Analytics, School of Computer Science, University College Dublin, Ireland
†School of Computer Science, University of Birmingham, United Kingdom
‡School of Computing and Mathematical Sciences, University of Leicester, United Kingdom
Email: honghui.du@ucd.ie, l.l.minku@bham.ac.uk, aonghus.lawlor@ucd.ie, hz143@leicester.ac.uk

*Abstract*—Label concepts in multi-label data streams often experience drift in non-stationary environments, either independently or in relation to other labels. Transferring knowledge between related labels can accelerate adaptation, yet research on multi-label transfer learning for data streams remains limited. To address this, we propose two novel transfer learning methods: BR-MARLENE leverages knowledge from different labels in both source and target streams for multi-label classification; BRPW-MARLENE builds on this by explicitly modelling and transferring pairwise label dependencies to enhance learning performance. Comprehensive experiments show that both methods outperform state-of-the-art multi-label stream approaches in non-stationary environments, demonstrating the effectiveness of inter-label knowledge transfer for improved predictive performance. The implementation is available at https://github.com/nino2222/MARLENE.

*Index Terms*—Concept drift, non-stationary environment, multi-source, multi-label, class imbalance, transfer learning.

## I. INTRODUCTION

Most research on data stream learning concentrates on streams with single labels [1]. However, many practical data streaming applications naturally adopt a multi-label paradigm, where each incoming data example has more than one label [2]. For example, a social media post could be tagged with several descriptors, or a movie might be classified under various predefined genres (e.g., Action, Crime, Historical), with each tag or genre representing a unique label.

A key challenge in multi-label data stream learning within non-stationary environments is that the stream typical contains multiple evolving label concepts, and drifts in one label's concept may occur synchronously or asynchronously with others [1], [3]. For example, a financial crisis may cause drift in labels like "economy", "employment", and "markets", while leaving "sports" or "entertainment" unaffected. Thus, while it is impractical to assume all labels drift together, it is also unrealistic to assume that drifts across labels are always independent [2]. Moreover, in practice, multi-label stream learning is often further complicated by class imbalance, and it is difficult to predict whether the positive or negative class dominates for each label [2]. Combined with evolving concept drift, this makes learning in non-stationary environments highly challenging.

Transfer learning has shown effectiveness by reusing knowledge from one learnt domain to accelerate learning in new domains [4]. When there is a relationship between the concepts of various labels, the knowledge learnt from one label can help improve learning of the new concept in another. Therefore, transferring such knowledge may be particularly beneficial for accelerating adaptation to concept drift in multi-label data streams. However, no research has yet investigated multi-label transfer learning for data streams.

Therefore, our paper aims to answer the following research questions: *How can knowledge transfer enhance predictive performance in multi-label learning under non-stationary environments? And how beneficial is it?*

To answer these questions and more effectively handle concept drift in multi-label data streams, we propose two novel multi-label online transfer learning approaches: Binary Relevance and Binary Relevance PairWise Multi-Label classification in non-stationary environments with Multi-Source Transfer Learning (BR-MARLENE and BRPW-MARLENE). We consider a multi-source scenario where both source and target streams may experience concept drift and class imbalance across labels. BR-MARLENE transfers useful knowledge from models trained on other labels to improve prediction for each target label. BRPW-MARLENE extends this by modelling and transferring knowledge across label pair dependencies to further enhance prediction (e.g., using knowledge from other label dependencies to predict the relationship between two given labels). Experiments show that both BR/BRPW-MARLENE outperform existing methods over time. BRPW-MARLENE achieves higher accuracy by modelling label dependencies, while BR-MARLENE offers faster execution.

The main contributions of this paper are:

- We propose two novel transfer learning approaches that transfers knowledge across labels and dependencies to enhance multi-label data stream prediction.
- We propose a novel weighting scheme that assigns each sub-classifier label-specific weights, prioritising correct predictions on minority and difficult cases to address class imbalance and enhance label-wise accuracy.
- We identify limitations in current multi-label stream evaluation metrics and propose new metrics by adapting G-Mean [5] to better reflect multi-label model performance under class imbalance.

## II. PROBLEM FORMULATION

Consider $D = \{(\mathbf{x}^t, \mathbf{y}^t)\}_{t=0}^{\infty}$ as a potentially infinite multi-label data stream; $\mathbf{x}^t \in \mathcal{X}$ are the input features of the example received at time step $t$; $\mathcal{X}$ is the input space; $\mathbf{y}^t = \{y_q^t\}_{q=1}^{|\mathcal{L}|} \in \mathcal{Y} = \{0,1\}^{|\mathcal{L}|}$, where set $\mathcal{L} = \{l_q\}_{q=1}^{|\mathcal{L}|}$ contains all possible labels of stream $D$, $y_q^t = 1$ if label $l_q$ is relevant to $\mathbf{x}^t$, and $y_q^t = 0$ otherwise. A concept for a specific $l_q$ can be defined as the conditional distribution $P(y_q|\mathbf{x})$. In non-stationary environments, the data stream $D$ may experience concept drift, which occurs when the distribution changes over time (i.e., there exist $t' \neq t''$ such that $P^{t'}(y_q|\mathbf{x}) \neq P^{t''}(y_q|\mathbf{x})$). It's worth noting that drift across different labels can occur either synchronously or asynchronously.

The objective of multi-label data stream learning is to train a predictive model $h(\mathbf{x}) : \mathcal{X} \to \mathcal{Y}$. Unlike chunk-based learning, this paper adopts an online paradigm that updates the model incrementally using only the previous model $h^{t-1}$ and the current data point $(\mathbf{x}^t, \mathbf{y}^t)$ to construct an updated model $h^t$ at each time step $t$. Compared with chunk-based approaches, online learning enables faster adaptation to drift and requires less memory, as it avoids storing large data chunks.

The domain of $D$ is denoted as $\mathcal{D} = \{\mathcal{X}, P(\mathbf{x})\}$, where $P(\mathbf{x})$ is the marginal probability distribution. We consider a scenario involving $n + 1$ data streams from various sources. Specifically, the data stream $D_i$ originates from the source $i \in \{S_1, S_2, \cdots, S_n, T\}$, where $D_{S_n}$ is the $n$th source data stream and $D_T$ is the target stream. For a target stream label, all other target and source stream labels are treated as sources. We investigate inductive transfer learning: $\mathcal{D}_{S_n} \neq \mathcal{D}_T$ or $\mathcal{D}_{S_n} = \mathcal{D}_T$, while $P(y_q|\mathbf{x}_i) \neq P(y_{q'}|\mathbf{x}_{i'})$.

## III. RELATED WORK

### A. Transfer Learning in Non-stationary Environments

Existing transfer learning in non-stationary environments mainly targets single-label data streams. Most of them are online inductive transfer approaches. DDD [6] and OWA [7] both transfer knowledge from previous to current concepts within the target stream: DDD leverages a diverse ensemble, while OWA uses ensemble weighting. Neither approach uses external source data. Melanie [8] is the first approach to transfer knowledge from multiple source streams by maintaining a weighted ensemble. OBAL [9] aligns the feature distributions of each source stream with that of the target stream, making different data streams share similar feature spaces and thereby reducing the negative impact of covariate shift on transfer learning. MARLINE [10] projects the target domain onto various source domains, allowing models trained on the source data to also make predictions on the target data, thus mitigating differences between the source and target domains.

There are also some chunk-based transfer approaches. DTEL [11] transfers the structure of a decision tree from an old data chunk to a new concept. CDTL [12] shares a similar idea with Melanie but uses a chunk-based ensemble. OTL-CE [13] addresses class evolution by identifying shared classes between source and target domains and building ensembles

for these classes. AOMSDA [14] leverages complementary information from multiple sources through a central moment discrepancy regularizer and employs node weighting to address covariate shift. However, as chunk-based methods, they cannot dynamically detect or adapt to changes in data streams.

All these transfer learning methods are designed for single-label data streams and cannot be directly applied to multi-label stream learning.

### B. Multi-Label Learning in Non-stationary Environments

Due to the flexibility and generality, problem transformation methods (e.g., Binary Relevance (BR) [15], Classifier Chain (CC) [16], Pairwise (PW) [17], Pruned Sets (PS) [18]) are the most widely used approach for multi-label learning. Problem transformation converts a multi-label problem into multiple single-label problems. Ensemble methods often incorporate these approaches as base classifiers [16], [18], [19]. However, these methods typically assume a stationary environment.

Some methods [20], [21] have been proposed for multi-label data stream learning in non-stationary environments, where problem transformation and ensemble strategies are also commonly applied. [22] proposes a max-relevance and min-redundancy based algorithm adaptation approach to handle high-dimensional features in non-stationary multi-label data streams. MLHAT [23] uses a Hoeffding adaptive tree to address these challenges by considering label relations and co-occurrences during tree partitioning, dynamically adapting leaf learners, and applying concept drift detection to quickly update underperforming branches. GOOWE-ML [3], which assigns confidence-based weights to each base classifier using sliding windows and least squares; it can be combined with any incremental multi-label classifier (e.g., GOBR, GOCC, GOPS, GORT) to improve performance. ADWIN Bagging [24] uses online bagging with ADWIN for concept drift detection, replacing poorly performing base models when drift is detected. For multi-label problems, it uses online multi-label base learners (such as Hoeffding Trees) and monitors a multi-label performance metric. Variants include EaBR, EaCC, and EaPS [25], which use BR, CC, and PS as base learners, respectively.

None of the above approaches leverage transfer learning to improve adaptation to new concepts, which could be especially valuable for multi-label data stream learning as discussed in Section I.

## IV. PROPOSED METHODS

### A. BR-MARLENE

BR-MARLENE takes one or more multi-label data streams as input (e.g., one target and others as sources), which may come from the same or different domains. Each data stream is processed label-wise. For each label, an independent binary sub-classifier is trained to learn its distribution and predict whether the label is relevant for a given instance. Concept drift is monitored for each label using drift detectors. When drift occurs, a new sub-classifier is trained for the updated concept and added to the ensemble, retaining both old and

new classifiers. Our approaches are designed without hyper-parameters to avoid both the need for tuning and the variability in performance. Only the base classifier and the drift detector need to be specified prior to training.

The core idea of BR-MARLENE is to improve predictive performance for a target stream by leveraging sub-classifiers trained on both source and target streams via transfer learning. It maintains an ensemble classifier comprising all sub-classifiers trained on different labels across different streams. For each target label, a unique set of weights is assigned to these sub-classifiers based on their performance on that label. Whenever a new target instance is available, BR-MARLENE allows all sub-classifiers to make predictions and applies different label-specific weight combinations to aggregate these predictions for each label, producing accurate results.

*1) Learning Process:* BR-MARLENE's learning process is shown in Algorithm 1. $\mathcal{M}$ denotes all received source and target streams. Following the binary relevance setting, upon receiving an example from stream $D_i$, BR-MARLENE creates $|\mathcal{L}_i|$ sub-classifiers and adds them to the ensemble $\mathcal{H}$, each trained with a specific label distribution. Any single-label binary classification method can serve as the sub-classifier (e.g., Hoeffding Tree [26]). For each target stream label, every sub-classifier initialises and maintains a set of performance indicators (detailed in Section IV-A2) to record its performance on this label, assess how well it matches the label's distribution, and update its weights in various label-specific weight sets (line 2 to 6).

Since concept drifts may occur on each label, we use a drift detection method (e.g., DDM-OCI [27]) to monitor changes per label. If a concept drift is detected for a label, BR-MARLENE generates a new sub-classifier to learn the upcoming new concept of that label and adds it to the ensemble (line 8 to 10), so the ensemble includes sub-classifiers representing various concepts for each label. The most recent sub-classifier for each label is used for drift monitoring. If drift occurs in a target stream label, all sub-classifiers in the ensemble reset their performance indicators for that label (line 11 to 12).

As mentioned, class imbalance is likely to occur for each label in multi-label data streams. Therefore, an integrated resampling strategy is adopted to increase the sampling rate of minority-class examples. For each sub-classifier, we record the number of positive and negative examples learnt ($n^+$ and $n^-$). When a training sub-example $(\mathbf{x}^t, y^t)$ is available, the training rate $k$ (i.e., the number of times the classifier is trained on this sub-example) is sampled from a Poisson distribution based on the ratio of majority to minority class counts as follows:

$$k \sim \begin{cases} Poisson(\frac{max(n^+,n^-)}{n^-}), & y = 0 \\ Poisson(\frac{max(n^+,n^-)}{n^+}), & y = 1 \end{cases} \qquad (1)$$

This ensures that minority-class examples are sampled more frequently during training (lines 14).

If the new training example is from the target stream, the weighting scheme in Section IV-A2 is used to update each sub-classifier's performance indicators and weights (line 16).

---

**Algorithm 1:** Learning Procedure of BR-MARLENE.

**Input:** $(\mathbf{x}_i^t, \mathbf{y}_i^t) \in D_i \quad i \in \{S_1, S_2, \cdots, S_n, T\}$

1 **while** *Receive a new example $(\mathbf{x}_i^t, \mathbf{y}_i^t)$* **do**
2    **if** $i \notin \mathcal{M}$ **then**
3      $\mathcal{M} \leftarrow \mathcal{M} \cup i$
4      **for** $l_q \in \mathcal{L}_i$ **do**
5        Initialise a sub-classifier $h$ for the label.
6        $\mathcal{H} \leftarrow \mathcal{H} \cup h$
7    **for** $l_q \in \mathcal{L}_i$ **do**
8      **if** $DriftDetection_{i,q}(\mathbf{x}_i^t, y_{i,q}^t) = True$ **then**
9        Initialise a new sub-classifier $h$ for the label.
10        $\mathcal{H} \leftarrow \mathcal{H} \cup h$
11        **if** $i = T$ **then**
12          Reset performance indicators of all sub-classifiers.
13    **for** $l_q \in \mathcal{L}_i$ **do**
14      Train the model as shown in Section IV-A1
15    **if** $i = T$ **then**
16      Update sub-classifiers' weights as shown in Section IV-A2

---

*2) Sub-classifiers' Weighting:* Each sub-classifier is assigned $|\mathcal{L}_T|$ weights to reflect its match with the distributions of different labels in the target stream, helping to avoid negative transfer and deal with concept drift. Standard ensemble weighting schemes often assign weights based on overall accuracy. In imbalanced environments, majority-class dominance allows sub-classifiers to earn high weights by performing well only on majority classes, even if they fail on minority classes. This leads the ensemble to neglect performance on minority-class predictions. Therefore, to better allocate each sub-classifier's contribution to the prediction, we propose a novel weighting scheme that rewards higher weights for correctly predicting difficult or minority class examples and assigns lower weights for easy or majority class cases.

For each sub-classifier, its binary classification results on a given target label is recorded using true positives ($TP$), false positives ($FP$), true negatives ($TN$), and false negatives ($FN$), which are updated incrementally as new examples arrive. Unlike offline learning, the full dataset is unavailable in advance; a class that appears to be the majority early on may later become the minority, making it impossible to foresee which classes are majority or minority. Consequently, we use dynamic correction factors (e.g., $\kappa^+$ and $\kappa^-$) to balance the impact of positive and negative examples. The contributions of positive and negative examples should be equal ($n^+ \times \kappa^+ = n^- \times \kappa^-$), and their sum should match the total number of examples (e.g., $n^+ \times \kappa^+ + n^- \times \kappa^- = n^+ + n^-$). Given this, the correction factors are defined as follows:

$$\kappa^+ \leftarrow \frac{n^+ + n^-}{2 \cdot n^+}, \quad \kappa^- \leftarrow \frac{n^+ + n^-}{2 \cdot n^-} \qquad (2)$$

Each sub-classifier's performance on a given label is measured by Positive Predictive Value ($PPV$) and Negative Predictive Value ($NPV$). By incorporating $\kappa^+$ and $\kappa^-$ into $PPV$ and $NPV$, we ensure each example's impact is balanced by class. The $PPV$ and $NPV$ can be formulated as follows:

$$PPV \leftarrow \frac{TP \cdot \kappa^+}{TP \cdot \kappa^+ + FP \cdot \kappa^-} \qquad (3)$$

$$NPV \leftarrow \frac{TN \cdot \kappa^-}{TN \cdot \kappa^- + FN \cdot \kappa^+} \qquad (4)$$

Through $PPV$ and $NPV$, we obtain the statistical probability that a sub-classifier's positive or negative prediction is correct. These serve as reliability measures for predictions and are used to calibrate the sub-classifier's output:

$$\hat{P}^+ \leftarrow P^+ \cdot PPV + P^- \cdot (1 - NPV) \qquad (5)$$

$$\hat{P}^- \leftarrow P^- \cdot NPV + P^+ \cdot (1 - PPV) \qquad (6)$$

where $\hat{P}^+$ aggregates the probability of a positive prediction ($P^+ = P(h(\mathbf{x}_T) = 1)$) weighted by its reliability ($PPV$) and the probability of a negative prediction ($P^- = P(h(\mathbf{x}_T) = 0)$) when the true class is actually positive ($1 - NPV$); similarly, $\hat{P}^-$ aggregates correct negative predictions ($P^- \cdot NPV$) and false positives ($P^+ \cdot (1 - PPV)$). By combining prediction reliability (e.g., $PPV$ and $NPV$) and correction factors (e.g., $\kappa^+$ and $\kappa^+$), the calibrated probabilities ($\hat{P}^+$ and $\hat{P}^-$) correct potential overconfidence in sub-classifier predictions under class imbalance and helps prevent predictions from being biased toward the majority class.

For a given label, an example is considered harder to classify (e.g., minority class example), the more the probability of an incorrect prediction exceeds that of a correct prediction. The weight $\frac{\lambda_{SW}}{\lambda_{SC}}$ for an example on the given label reflects the ensemble's classification difficulty for that example and is computed as follows:

$$\lambda_{SC} \leftarrow \sum_{h \in \mathcal{H}} \hat{P}^y(h(\mathbf{x}_T)), \quad \lambda_{SW} \leftarrow \sum_{h \in \mathcal{H}} \hat{P}^{\overline{y}}(h(\mathbf{x}_T)) \qquad (7)$$

where $\hat{P}^y(h(\mathbf{x}_T))$ is the calibrated probability that sub-classifier $h$ predicts the true class $y$; $\hat{P}^{\overline{y}}(h(\mathbf{x}_T))$ is the calibrated probability that it predicts the incorrect class. The easier the ensemble classifies an example, the smaller its weight; the harder, the larger its weight.

For a given label, each sub-classifier is weighted by its performance on that label. SC and SW are the sub-classifier's correct and incorrect prediction scores, each consisting of the example's weight (reflecting classification difficulty) and the sub-classifier's relative contribution to the ensemble's prediction. SC and SW are updated incrementally as new examples arrive. The weight $\alpha$ is computed for each sub-classifier as follows:

$$SC^t \leftarrow SC^{t-1} + \frac{\lambda_{SW}}{\lambda_{SC}} \cdot \frac{\hat{P}^y(h(\mathbf{x}_T))}{\lambda_{SC}} \qquad (8)$$

$$SW^t \leftarrow SW^{t-1} + \frac{\lambda_{SW}}{\lambda_{SC}} \cdot \frac{\hat{P}^{\overline{y}}(h(\mathbf{x}_T))}{\lambda_{SW}} \qquad (9)$$

$$\alpha \leftarrow \frac{SC}{SC + SW} \qquad (10)$$

where $\frac{\hat{P}^y(h(\mathbf{x}_T))}{\lambda_{SC}}$ and $\frac{\hat{P}^{\overline{y}}(h(\mathbf{x}_T))}{\lambda_{SW}}$ quantify the sub-classifier's proportion among all correct and incorrect predictions, respectively, for the current example in the ensemble. Thus, when a sub-classifier predicts correctly while most others are incorrect, it receives a higher $SC$ and a lower $SW$.

*3) Voting Procedure:* BR-MARLENE uses the same set of sub-classifiers for all labels, but assigns a different weight combination for each label. The ensemble $\mathcal{H}$'s vote for instance $\mathbf{x}_T$ on a given target label $l_{T,q}$ is computed as follows:

$$P_q(\mathcal{H}(\mathbf{x}_T)) = \sum_{h \in \mathcal{H}} \alpha_h^q \cdot \hat{P}(h(\mathbf{x}_T)) \qquad (11)$$

$$\hat{y}_{T,q} = \arg \max_{y \in \{0,1\}} P_q(\mathcal{H}(\mathbf{x}_T)) \qquad (12)$$

### B. BRPW-MARLENE

While Binary Relevance (BR) is effective and widely used, its assumption of label independence overlooks potential dependencies between labels [28]. It may yield less accurate predictions when labels are correlated, as valuable label co-occurrence information is ignored. To address the limitations of BR, we propose BRPW-MARLENE, which enhances BR-MARLENE's predictive performance by modelling pairwise label dependencies and enabling transfer learning between PW-classifiers for dependency prediction. For clarity, we refer to BR-MARLENE's sub-classifiers as BR-classifiers and those of BRPW-MARLENE as PW-classifiers in this section; both contribute to BRPW-MARLENE's final predictions.

*1) Learning Process:* BRPW-MARLENE first runs the standard BR-MARLENE training to generate BR-classifiers for each label. In addition, it constructs $|\mathcal{L}|(|\mathcal{L}| - 1)$ PW-classifiers for each possible pair of labels in every data stream $D_i$. Each PW-classifier is trained to model the dependency between a specific pair of labels (e.g., $l_q$ and $l_{q'}$), using input features composed of the original features and one label ($\mathbf{x}, y_q$), and predicting the other label $y_{q'}$ in the pair. The training process for PW-classifiers is the same as that of BR-classifiers, including concept drift detection and adaptive model updates. When a drift is detected in a dependency, a new PW-classifier is added to represent the new concept. Each PW-classifier maintains dependency-specific performance indicators and weights, using a weighting scheme similar to BR-classifiers, but specifically targeting predictive performance for each label pair dependency. This setup enables BRPW-MARLENE to learn and adapt to evolving pairwise label dependencies in the data streams.

*2) Voting Procedure:* To generate the prediction for a target label, BRPW-MARLENE first computes the BR-MARLENE ensemble prediction for each label, which aggregates the outputs of all BR-classifiers for that label. These initial predictions are then used as additional inputs to predict the dependencies between each pair of labels $l_{T,q}$ and $l_{T,q'}$ as:

$$P_{q,q'}(\mathcal{H}^{PW}(\mathbf{x}_T, \hat{y}_{T,q})) = \sum_{h \in \mathcal{H}^{PW}} \alpha_h^{q,q'} \cdot \hat{P}(h(\mathbf{x}_T, \hat{y}_{T,q}))$$

(13)

where $\hat{y}_{T,q}$ is BR-MARLENE's prediction. The final prediction for label $l_{T,q}$ is obtained by combining the outputs of BR-classifiers and PW-classifiers:

$$P_q(\mathcal{H}^{BRPW}(\mathbf{x}_T))$$
$$= \sum_{q'=1, q' \neq q}^{|\mathcal{L}_T|} P(y_{T,q'}) P(y_{T,q}|y_{T,q'}) + P(y_{T,q}|\mathbf{x}_T)$$

(14)

$$\hat{y}_{T,q} = \arg \max_{y \in \{0,1\}} P_q(\mathcal{H}^{BRPW}(\mathbf{x}_T))$$

(15)

where $P(y_{T,q'}) = P_{q'}(\mathcal{H}^{BR}(\mathbf{x}_T))$, $P(y_{T,q'})P(y_{T,q}|y_{T,q'}) = P_{q',q}(\mathcal{H}^{PW}(\mathbf{x}_T, \hat{y}_{T,q'}))$, $P(y_{T,q}|\mathbf{x}_T) = P_q(\mathcal{H}^{BR}(\mathbf{x}_T))$.

## V. TIME COMPLEXITY ANALYSIS

Since PW-classifiers use the same algorithm as BR-classifiers, both take $\mathcal{O}(f_h^{train})$ for a single training, $\mathcal{O}(f_h^{pred})$ for prediction, and drift detection takes $\mathcal{O}(f_{DD})$.

When the current example is from the target stream, the overall time complexity of BR-MARLENE's learning procedure is $\mathcal{O}(|\mathcal{L}_T| \times f_{DD} + \sum_{q=1}^{|\mathcal{L}_T|} k_{T,q} \times f_h^{train} + |\mathcal{L}_T| \times |\mathcal{H}^{BR}| \times f_h^{pred})$, where $k_{T,q}$ is the integer from the corresponding Poisson distribution, and $\sum_{q=1}^{|\mathcal{L}_T|} k_{T,q}$ is the total number of sub-classifier training operations; The overall BRPW-MARLENE training complexity is $\mathcal{O}(|\mathcal{L}_T|^2 \times f_{DD} + (\sum_{q=1}^{|\mathcal{L}_T|} k_{T,q} + \sum_{q=1}^{|\mathcal{L}_T|} \sum_{q'=1, q' \neq q}^{|\mathcal{L}_T|} k_{T,q,q'}) \times f_h^{train} + (|\mathcal{H}^{BR}| + (|\mathcal{L}_T| - 1) \times |\mathcal{H}^{PW}|) \times |\mathcal{L}_T| \times f_h^{pred})$.

When learning from a source stream $S_n$, BR/PW sub-classifier weights are not updated, resulting in lower training time complexity: BR-MARLENE, $\mathcal{O}(|\mathcal{L}_{S_n}| \times f_{DD} + \sum_{q=1}^{|\mathcal{L}_{S_n}|} k_{S_n,q} \times f_h^{train})$; BRPW-MARLENE, $\mathcal{O}(|\mathcal{L}_{S_n}|(|\mathcal{L}_{S_n}| - 1) \times f_{DD} + \sum_{q=1}^{|\mathcal{L}_{S_n}|} \sum_{q'=1, q' \neq q}^{|\mathcal{L}_{S_n}|} k_{S_n,q,q'} \times f_h^{pred})$.

In BR-MARLENE's prediction procedure, each sub-classifier predicts the relevance of each target label, giving an overall prediction time complexity of $\mathcal{O}(|\mathcal{L}_T| \times |\mathcal{H}^{BR}| \times f_h)$. For BRPW-MARLENE, predictions from all BR-classifiers and PW-classifiers are combined, with an overall time complexity of $\mathcal{O}((|\mathcal{H}^{BR}| + (|\mathcal{L}_T| - 1) \times |\mathcal{H}^{PW}|) \times |\mathcal{L}_T| \times f_h)$

## VI. EXPERIMENTAL SETUP

### A. Datasets

*1) Real-World Datasets:* We selected seven commonly used real-world datasets [3], [19], [25], [29] to evaluate our proposed methods. To better understand these datasets and the experimental results on them, it is important to measure the degree of multi-label in each. Label density ($LDen$) [30] is a

TABLE I: Tabulation of real-world datasets; superscript indicates input feature type: binary ($b$) or numeric ($n$).

| Dataset($D$) | $\lvert D\rvert$ | $\lvert\mathbf{x}_D\rvert$ | $\lvert\mathcal{L}_D\rvert$ | $LDen(D)$ | $LIR(D)$ | $LSIR(D)$ |
|---|---|---|---|---|---|---|
| Slashdot$^b$ | 3782 | 1079 | 22 | 0.054 | 0.054 | 0.054 |
| Ohsumed$^b$ | 13929 | 1002 | 23 | 0.072 | 0.072 | 0.072 |
| Reuters$^n$ | 6000 | 500 | 103 | 0.014 | 0.014 | 0.014 |
| Yeast$^n$ | 2417 | 103 | 14 | 0.303 | 0.232 | 0.297 |
| 20NG$^b$ | 19300 | 1006 | 20 | 0.051 | 0.051 | 0.051 |
| TMC2007$^b$ | 28596 | 500 | 22 | 0.098 | 0.092 | 0.098 |
| IMDB$^b$ | 120919 | 1001 | 28 | 0.071 | 0.071 | 0.071 |

widely used measure reflecting the average number of labels associated with each instance and can be computed as follows:

$$LDen(D) = \frac{1}{|D|} \sum_{t=1}^{|D|} \sum_{q=1}^{|\mathcal{L}_D|} \frac{y_q^t}{|\mathcal{L}_D|}$$

(16)

As class imbalance frequently occurs in multi-label data streams, the imbalance rate is an important measure to consider. In single-label binary problems, the Imbalance Rate ($IR$) refers to the occurrence probability of the minority class. For multi-label problems, we define the Label Imbalance Rate ($LIR$) as the mean $IR$ across all labels, and the Label-Set Imbalance Rate ($LSIR$) as the mean $IR$ across all label sets for each example:

$$LIR(D) = \frac{\sum_{q=1}^{|\mathcal{L}_D|} N_q^{min}}{|\mathcal{L}_D| \cdot |D|}, \quad LSIR(D) = \frac{\sum_{t=1}^{|D|} N_{\mathbf{y}^t}^{min}}{|\mathcal{L}_D| \cdot |D|}$$

(17)

where $N_q^{min}$ is the number of samples in the minority class for the $q$-th label, and $N_{\mathbf{y}^t}^{min}$ is the number of samples in the minority class for the label set of the $t$-th example. $LIR$ measures average imbalance across labels, while $LSIR$ captures imbalance within each instance's label set; lower values mean greater imbalance. If the minority class for every label in $D$ is positive, then $LIR(D) = LDen(D)$; likewise, if the minority class for each example's label set is positive, $LSIR(D) = LDen(D)$. Table I summarises each dataset's parameters and measurements. Each dataset includes only a target stream without source streams. This setting allows us to analyse the benefit of transfer between label concepts within a stream, which is the main focus of this work.

*2) Synthetic Datasets:* Due to privacy concerns, few real-world multi-label datasets are available, and traditional offline datasets are unsuitable for non-stationary environments [31], making it difficult to find appropriate source data for each real-world dataset. To further investigate the impact of concept drift and multiple sources, we generate synthetic datasets composed of several synthetic streams (sources and target).

Each dataset consists of two numeric features and five binary labels ($|\mathcal{L}_D| = 5$). For each source or target, input features follow a mixture of five Gaussian distributions. Each class of a label is linked to specific Gaussians, with one representing the positive class and the other four representing negatives. Target datasets vary in size per Gaussian (50, 500, 5000), simulating small, medium and large samples. Each

target has both similar and non-similar source datasets to generate diverse synthetic streams. Datasets with no source represent transfer only between labels within the same target stream. This setup allows us to analyse 1) the benefit of transfer from source to target labels, 2) transfer between labels within the target stream. All sources have 5000 examples per Gaussian distribution. To better assess the proposed methods and their ability to handle concept drift on each label, two target labels ($l_1$ and $l_2$) are evaluated under three scenarios: stable (S), abrupt drift (A), and incremental drift (I), resulting in six possible combinations (e.g., IA: $l_1$ incremental, $l_2$ abrupt drift). Datasets are named according to the drift type of $l_1$ and $l_2$ (e.g., IA: $l_1$ has incremental drift, $l_2$ has abrupt drift; AS: $l_1$ has abrupt drift, $l_2$ is stable). The other three labels remain stable.

### B. Benchmark Methods

We select 11 SOTA multi-label approaches available in the Massive Online Analysis (MOA) tool [32] via MEKA [33], covering all major groups of existing multi-label methods: EBR [16], ECC [16], EPS [18], ERT [19], EaBR [25], EaCC [25], EaPS [25], GOBR [3], GOCC [3], GOPS [3], GORT [3]. EaBR, EaCC, and EaPS have built-in ADWIN [34] for concept drift detection, while BR-MARLENE uses the class imbalance concept drift detector (DDM-OCI [27]).

We use sliding window-based prequential evaluation [35] with a window size of 10% of $|D|$. As GOBR, GOCC, GOPS, and GORT are chunk-based and can only predict after the first chunk is filled, evaluation for all methods begins when these approaches are ready to make predictions. All approaches use Hoeffding Trees [26] as the base classifier, with an ensemble size of 10 (i.e., 10 sub-classifiers) except for BR-MARLENE and BRPW-MARLENE, whose ensemble sizes depend on the number of possible labels in the dataset.

Thirty runs are performed for all stochastic approaches, while GOBR, GOCC, GOPS, and GORT, being deterministic, require only a single run. The average Macro/Micro/Label-set-based G-Mean and computation time over 30 runs are reported.

### C. Evaluation Metrics

In the multi-label setting, simple metrics like accuracy do not adequately capture model performance. Multi-label classifiers are typically evaluated using either label-set-based or label-based measures. Label-set-based measures (e.g., Hamming Score (HScore) and Hamming Loss (HLoss) [36]) assess model performance by averaging predictions over each example's label set; HScore is the mean fraction of correct labels per instance, while HLoss is the mean proportion of misclassified labels. However, both metrics may be misleading in imbalanced datasets because they fail to capture performance on rare labels, similar to the limitations of accuracy in single-label imbalance.

Unlike label-set-based measures, label-based measures are calculated by averaging the model's performance on each individual label (using micro or macro averaging) across all examples [37]:

$$M_{marco} = \frac{\sum_{q=1}^{|\mathcal{L}_D|} M(TP_q, FP_q, FN_q)}{|\mathcal{L}_D|} \quad (18)$$

$$M_{micro} = M(\sum_{q=1}^{|\mathcal{L}_D|} TP_q, \sum_{q=1}^{|\mathcal{L}_D|} FP_q, \sum_{q=1}^{|\mathcal{L}_D|} FN_q) \quad (19)$$

where $M$ is a performance metric (e.g., Recall, Precision, F-Score). Macro measures calculate the metric on each label and then average the results, making them sensitive to label imbalance. Micro measures, in contrast, aggregate counts across labels before computing the metric, thus reflecting overall performance and diluting the influence of individual label imbalance [38]. Recall, Precision, and F-Score are the most common metrics for macro and micro evaluation; as seen from equations 18 and 19, they focus on positive prediction performance and generally assume the positive class is the minority, which is not always the case (e.g., In Yeast, positives are the majority for two labels, and in TMC2007, one label also has majority positives, so $LIR \neq LDen$ in both datasets). A *dummy* classifier that always predicts the majority class for each label can still achieve high macro and micro scores, highlighting the limitations of these metrics under class imbalance.

To address the limitations of current metrics, we adapt the widely used G-Mean [5] from single-label class imbalance learning for use in multi-label settings. G-Mean evaluates model performance by taking the geometric mean of recall for both positive and negative classes [39]:

$$G - Mean = \sqrt[2]{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \quad (20)$$

where $\frac{TP}{TP+FN}$ is the positive class recall and $\frac{TN}{TN+FP}$ is the negative class recall. Macro and micro G-Mean are obtained by using G-Mean as the performance metric in Equations 18 and 19. Unlike recall, precision, or F-Score, G-Mean offers a more balanced assessment of model performance across both classes for each label.

Label-set-based G-Mean (LS-G-Mean) is computed by calculating $TP$, $FP$, $TN$, and $FN$ on each received label set:

$$G - Mean_{LS} =$$
$$\frac{1}{|D|} \sum_{t=1}^{|D|} G - Mean(TP^t, FP^t, TN^t, FN^t) \quad (21)$$

where $TP^t$, $FP^t$, $TN^t$, and $FN^t$ are the confusion matrix counts computed for the predicted $\hat{\mathbf{y}}^t$ and true label sets $\mathbf{y}^t$. Compared to HScore and HLoss, LS-G-Mean gives equal weight to all classes across label sets, helping to avoid the effects of class imbalance.

## VII. Experimental Results and Analysis

### A. Results on Real-World Datasets

The predictive performance of each algorithm on each dataset is shown in Table II. Due to the high computa-

TABLE II: Experimental results on each dataset. BR-M refers to BR-MARLENE, BRPW-M refers to BRPW-MARLENE, Macro refers to Macro-G-Mean, Micro to Micro-G-Mean, LS to Label-set-based-G-Mean, and Rank to Friedman's rank. Friedman's p-values are always $< 2.2 \times 10^{-16}$. The best approach is highlighted in red, and approaches without a significant difference according to the post-hoc Nemenyi test are shown in bold.

| Dataset | Slashdot | | Ohsumed | | Reuters | | Yeast | | 20NG | | TMC2007 | | IMDB | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metrics | Macro | Rank | Macro | Rank | Macro | Rank | Macro | Rank | Macro | Rank | Macro | Rank | Macro | Rank | Avg. Rank |
| **BRPW-M** | - | - | - | - | - | - | 0.567 | - | - | - | - | - | - | - | - |
| **BR-M** | **0.357** | **1.116** | **0.571** | **1.019** | **0.415** | **1.003** | **0.539** | **1.012** | **0.699** | **1.023** | **0.752** | **1.003** | **0.411** | **1.000** | **1.025** |
| EBR | 0.023 | 7.850 | 0.294 | 3.662 | 0.048 | 5.877 | 0.232 | 6.464 | 0.506 | 3.075 | 0.491 | 2.203 | 0.059 | 4.271 | 4.772 |
| ECC | 0.020 | 8.992 | 0.273 | 4.828 | 0.041 | 6.757 | 0.239 | 5.802 | 0.490 | 4.587 | 0.479 | 3.172 | 0.035 | 6.441 | 5.797 |
| EPS | 0.081 | 6.356 | 0.151 | 6.073 | 0.035 | 6.642 | 0.295 | 2.542 | 0.268 | 7.138 | 0.245 | 6.443 | 0.008 | 9.412 | 6.372 |
| ERT | 0.000 | 10.290 | 0.030 | 10.477 | 0.000 | 10.718 | 0.000 | 11.991 | 0.112 | 11.101 | 0.006 | 12.000 | 0.002 | 11.273 | 11.121 |
| EaBR | 0.010 | 8.776 | 0.227 | 5.893 | 0.020 | 8.501 | 0.231 | 6.698 | 0.464 | 5.095 | 0.436 | 3.979 | 0.031 | 6.789 | 6.533 |
| EaCC | 0.019 | 9.313 | 0.016 | 11.278 | 0.003 | 10.123 | 0.238 | 6.077 | 0.270 | 7.713 | 0.411 | 4.946 | 0.006 | 10.442 | 8.556 |
| EaPS | 0.110 | 3.996 | 0.126 | 7.327 | 0.049 | 4.812 | 0.288 | 3.598 | 0.199 | 9.364 | 0.246 | 6.345 | 0.012 | 9.424 | 6.409 |
| GOBR | 0.027 | 7.143 | 0.126 | 7.293 | 0.042 | 5.700 | 0.221 | 7.178 | 0.294 | 6.355 | 0.177 | 7.979 | 0.126 | 2.751 | 6.343 |
| GOCC | 0.028 | 7.106 | 0.134 | 6.949 | 0.013 | 9.327 | 0.196 | 9.046 | 0.176 | 9.779 | 0.163 | 9.052 | 0.154 | 2.249 | 7.644 |
| GOPS | 0.196 | 2.415 | 0.196 | 4.558 | 0.047 | 4.704 | 0.218 | 7.358 | 0.271 | 6.523 | 0.138 | 9.878 | 0.014 | 8.685 | 6.303 |
| GORT | 0.068 | 4.647 | 0.092 | 8.642 | 0.060 | 3.837 | 0.177 | 10.233 | 0.293 | 6.247 | 0.090 | 11.000 | 0.043 | 5.263 | 7.124 |
| Metrics | Micro | Rank | Micro | Rank | Micro | Rank | Micro | Rank | Micro | Rank | Micro | Rank | Micro | Rank | Avg. Rank |
| **BRPW-M** | - | - | - | - | - | - | 0.663 | - | - | - | - | - | - | - | - |
| **BR-M** | 0.572 | 2.362 | **0.610** | **1.461** | **0.753** | **1.000** | 0.644 | 8.694 | **0.759** | **1.004** | **0.774** | **1.763** | 0.524 | 2.611 | **2.699** |
| EBR | 0.065 | 8.692 | 0.375 | 5.342 | 0.188 | 6.292 | **0.724** | **2.165** | 0.561 | 3.131 | 0.761 | 2.379 | 0.226 | 8.034 | 5.148 |
| ECC | 0.059 | 9.715 | 0.360 | 6.524 | 0.177 | 7.325 | 0.718 | 2.593 | 0.547 | 4.652 | 0.754 | 3.530 | 0.111 | 9.667 | 6.287 |
| EPS | 0.333 | 5.076 | 0.458 | 5.410 | 0.335 | 5.654 | 0.697 | 5.883 | 0.462 | 5.835 | 0.691 | 6.851 | 0.414 | 5.501 | 5.744 |
| ERT | 0.003 | 10.817 | 0.165 | 10.495 | 0.001 | 10.813 | 0.000 | 12.000 | 0.255 | 10.094 | 0.039 | 12.000 | 0.013 | 11.565 | 11.112 |
| EaBR | 0.040 | 9.572 | 0.309 | 7.727 | 0.112 | 9.198 | **0.724** | **2.445** | 0.523 | 5.148 | 0.755 | 2.784 | 0.133 | 9.327 | 6.600 |
| EaCC | 0.057 | 9.987 | 0.037 | 11.626 | 0.024 | 10.199 | 0.717 | 2.951 | 0.337 | 8.172 | 0.745 | 4.571 | 0.021 | 11.407 | 8.416 |
| EaPS | 0.287 | 5.975 | 0.415 | 7.070 | 0.380 | 3.882 | 0.700 | 5.015 | 0.358 | 8.655 | 0.682 | 6.480 | 0.415 | 5.837 | 6.131 |
| GOBR | **0.598** | **1.877** | 0.546 | 2.933 | 0.407 | 4.309 | 0.576 | 9.246 | 0.448 | 6.239 | 0.554 | 8.924 | 0.570 | 2.154 | 5.097 |
| GOCC | 0.155 | 7.333 | 0.256 | 9.165 | 0.081 | 9.448 | 0.577 | 9.645 | 0.230 | 10.871 | 0.531 | 10.439 | 0.372 | 5.683 | 8.941 |
| GOPS | 0.365 | 4.564 | 0.414 | 6.567 | 0.282 | 6.715 | 0.576 | 9.468 | 0.337 | 8.578 | 0.521 | 10.535 | 0.429 | 4.921 | 7.335 |
| GORT | **0.596** | **2.031** | 0.506 | 3.680 | 0.408 | 3.164 | 0.633 | 7.895 | 0.487 | 5.620 | 0.640 | 7.745 | **0.612** | **1.295** | 4.490 |
| Metrics | LS | Rank | LS | Rank | LS | Rank | LS | Rank | LS | Rank | LS | Rank | LS | Rank | Avg. Rank |
| **BRPW-M** | - | - | - | - | - | - | 0.636 | - | - | - | - | - | - | - | - |
| **BR-M** | 0.412 | 2.345 | **0.497** | **1.347** | **0.640** | **1.000** | 0.606 | 8.404 | **0.650** | **1.012** | **0.733** | **2.120** | 0.397 | 2.615 | **2.692** |
| EBR | 0.015 | 8.559 | 0.234 | 5.321 | 0.097 | 6.290 | **0.697** | **1.912** | 0.363 | 3.157 | 0.720 | 2.231 | 0.079 | 8.029 | 5.071 |
| ECC | 0.013 | 9.667 | 0.221 | 6.509 | 0.089 | 7.188 | 0.685 | 2.902 | 0.347 | 4.732 | 0.707 | 3.835 | 0.018 | 9.765 | 6.371 |
| EPS | 0.134 | 5.050 | 0.275 | 5.242 | 0.157 | 5.431 | 0.646 | 6.045 | 0.223 | 5.859 | 0.630 | 6.925 | 0.261 | 5.361 | 5.702 |
| ERT | 0.000 | 10.828 | 0.050 | 10.461 | 0.000 | 10.813 | 0.000 | 12.000 | 0.082 | 10.120 | 0.004 | 12.000 | 0.000 | 11.600 | 11.117 |
| EaBR | 0.006 | 9.832 | 0.178 | 7.743 | 0.046 | 9.116 | 0.697 | 2.374 | 0.332 | 5.075 | 0.717 | 2.516 | 0.030 | 9.233 | 6.556 |
| EaCC | 0.012 | 9.942 | 0.005 | 11.629 | 0.004 | 10.170 | 0.684 | 2.922 | 0.152 | 8.182 | 0.700 | 4.559 | 0.001 | 11.372 | 8.396 |
| EaPS | 0.101 | 5.969 | 0.230 | 6.953 | 0.205 | 3.448 | 0.652 | 5.169 | 0.138 | 8.760 | 0.619 | 6.495 | 0.262 | 5.707 | 6.071 |
| GOBR | **0.436** | **1.923** | 0.381 | 3.084 | 0.216 | 4.729 | 0.517 | 8.830 | 0.224 | 6.285 | 0.436 | 8.910 | 0.451 | 2.235 | 5.142 |
| GOCC | 0.031 | 7.336 | 0.087 | 9.177 | 0.015 | 9.557 | 0.517 | 9.658 | 0.057 | 10.887 | 0.405 | 10.416 | 0.208 | 6.020 | 9.007 |
| GOPS | 0.156 | 4.565 | 0.223 | 6.750 | 0.112 | 6.738 | 0.514 | 10.046 | 0.126 | 8.367 | 0.392 | 10.549 | 0.284 | 4.740 | 7.393 |
| GORT | **0.426** | **1.984** | 0.321 | 3.785 | 0.206 | 3.521 | 0.594 | 7.739 | 0.267 | 5.565 | 0.570 | 7.445 | **0.515** | **1.322** | 4.480 |

tional cost from modelling pairwise dependencies, BRPW-MARLENE is only evaluated on the Yeast dataset ($|\mathcal{L}| = 14$). As BRPW-MARLENE is an extension of BR-MARLENE and only tested on Yeast, we first compare BR-MARLENE with other benchmarks, and then compare BRPW-MARLENE with BR-MARLENE.

BR-MARLENE achieves the best average Friedman rank and significantly higher Macro-G-Mean values than all other approaches across datasets. Since Macro-G-Mean is particularly sensitive to minority class results of each label (as discussed in Section VI-C) and all datasets have very low $LIR$ (see Table I), this demonstrates that BR-MARLENE is especially effective at handling class imbalance for each label.

BR-MARLENE achieves the best average Friedman ranking in Micro-G-Mean across all datasets, reflecting strong global performance over all classes and labels. While it does not have the top Micro-G-Mean on three datasets (Slashdot, Yeast, and IMDB) because it assigns less weight to the majority class, it still delivers competitive results (e.g., BR-MARLENE's Micro-G-Mean on Slashdot is 0.572, compared to 0.598 for the best method GOBR). Results for LS-G-Mean are similar: BR-MARLENE has the best average Friedman ranking and

the highest ranking position on most datasets.

The average runtime for each approach is shown in Table III. BR-MARLENE has the shortest execution time on all datasets except Reuters, TMC2007, and IMDB, as these datasets have more labels and may contain more concept drift, increasing the number of sub-classifiers. Nevertheless, BR-MARLENE remains the second fastest on TMC2007 and third fastest on Reuters and IMDB.

Comparing BR-MARLENE and BRPW-MARLENE on Yeast, BRPW-MARLENE achieves higher performance across all three evaluation metrics, demonstrating that modelling label dependencies can indeed improve performance. However, BRPW-MARLENE is nearly 53 times slower (see Table III) than BR-MARLENE due to the large number of pairwise models. More comparisons between BR-MARLENE and BRPW-MARLENE on synthetic datasets are discussed in the next section.

### B. Results on Synthetic Datasets

To further investigate the impact of multiple sources and different concept drift types, we run BR/BRPW-MARLENE on several synthetic datasets. The Friedman ranks for each

TABLE III: Tabulations of execution time. All values are in milliseconds. The best values of each column are in red.

| Dataset | Slashdot | Ohsumed | Reuters | Yeast | 20NG | TMC2007 | IMDB |
|---|---|---|---|---|---|---|---|
| **BRPW-M** | - | - | - | 82604 | - | - | - |
| **BR-M** | 24111 | 110207 | 88068 | 1562 | 134837 | 105846 | 5754256 |
| EBR | 400567 | 1932007 | 1958737 | 20823 | 2284106 | 2000631 | 31679670 |
| ECC | 400471 | 1804186 | 1947755 | 20072 | 2269085 | 1994518 | 31296078 |
| EPS | 64995 | 185806 | 34768 | 3961 | 273480 | 78551 | 296169 |
| ERT | 53813 | 238408 | 186590 | 8090 | 259720 | 225389 | 1895433 |
| EaBR | 394570 | 2049120 | 2100270 | 32151 | 2560537 | 2427817 | 22042432 |
| EaCC | 391400 | 1757911 | 2011307 | 32178 | 2214708 | 2337822 | 18848438 |
| EaPS | 169128 | 424801 | 112571 | 8383 | 765386 | 243306 | 639729 |
| GOBR | 559366 | 2001526 | 2024646 | 22191 | 2640190 | 2179204 | 26444018 |
| GOCC | 565521 | 2015383 | 2221846 | 25066 | 2643139 | 2281488 | 27882590 |
| GOPS | 113790 | 269166 | 76479 | 7536 | 432014 | 141485 | 546422 |
| GORT | 75365 | 294842 | 209673 | 7824 | 367842 | 334401 | 3200813 |

TABLE IV: Friedman's ranks on synthetic datasets: Macro, Micro, and LS refer to Macro-G-Mean, Micro-G-Mean, and Label-set-based-G-Mean, respectively. Best ranks are in red, bold indicates no significant difference by Nemenyi test, and all Friedman's p-values are always $< 2.2 \times 10^{-16}$

| Dataset | Size | BR w/o S | BRPW w/o S | BR w/ NS | BRPW w/ NS | BR w/ SS | BRPW w/ SS |
|---|---|---|---|---|---|---|---|
| Drift Type | Size | Macro | | | | | |
| SS | 50 | 4.392 | **2.768** | 4.292 | **2.926** | 4.152 | **2.470** |
| | 500 | 5.315 | 2.501 | 5.114 | 2.981 | 3.308 | **1.781** |
| | 5000 | 5.038 | 1.858 | 5.752 | 2.692 | 4.092 | **1.568** |
| IS | 50 | 4.522 | 2.966 | 4.853 | 2.748 | 3.796 | **2.115** |
| | 500 | 5.155 | 2.345 | 5.616 | 2.716 | 3.664 | **1.503** |
| | 5000 | 4.714 | **1.765** | 5.444 | 2.581 | 4.470 | 2.025 |
| II | 50 | 5.212 | 2.808 | 5.171 | 2.245 | 3.986 | **1.578** |
| | 500 | 5.081 | 3.094 | 5.323 | 2.189 | 3.802 | **1.512** |
| | 5000 | 4.853 | 2.847 | 5.214 | 1.893 | 4.579 | **1.614** |
| IA | 50 | 5.038 | 2.965 | 5.133 | 2.884 | 3.682 | **1.298** |
| | 500 | 4.877 | 2.619 | 5.216 | 2.461 | 4.235 | **1.591** |
| | 5000 | 4.641 | 2.501 | 5.248 | 1.975 | 4.852 | **1.783** |
| AA | 50 | 4.712 | 2.977 | 4.539 | **2.456** | 3.779 | **2.537** |
| | 500 | 5.104 | 2.289 | 4.817 | **1.613** | 4.514 | 2.064 |
| | 5000 | 4.996 | **1.793** | 5.291 | 2.233 | 4.525 | 2.162 |
| AS | 50 | 4.228 | 3.255 | 4.374 | 3.200 | 3.507 | **2.436** |
| | 500 | 5.084 | **2.380** | 4.994 | 3.070 | 3.172 | **2.300** |
| | 5000 | 4.955 | 1.957 | 5.860 | **1.691** | 4.038 | 2.499 |
| Avg. Rank | | 4.884 | 2.571 | 5.125 | 2.475 | 4.008 | **1.935** |
| Drift Type | Size | Micro | | | | | |
| SS | 50 | 4.380 | **2.804** | 4.280 | **2.910** | 4.094 | **2.532** |
| | 500 | 5.363 | 2.846 | 5.091 | 2.579 | 3.193 | **1.928** |
| | 5000 | 4.891 | 2.376 | 5.698 | **1.724** | 4.298 | 2.013 |
| IS | 50 | 4.226 | 3.244 | 4.538 | 3.032 | 3.704 | **2.256** |
| | 500 | 5.065 | 2.648 | 5.533 | 2.757 | 3.521 | **1.477** |
| | 5000 | 4.767 | **1.986** | 5.469 | 2.353 | 4.447 | **1.978** |
| II | 50 | 4.996 | 2.979 | 4.986 | 2.407 | 3.964 | **1.668** |
| | 500 | 4.706 | 3.239 | 5.087 | 2.270 | 4.252 | **1.445** |
| | 5000 | 4.802 | 2.953 | 5.090 | 1.810 | 4.766 | **1.578** |
| IA | 50 | 4.943 | 3.002 | 5.189 | 2.988 | 3.591 | **1.288** |
| | 500 | 4.651 | 2.806 | 5.058 | 2.807 | 4.149 | **1.529** |
| | 5000 | 4.561 | 2.640 | 5.222 | 1.896 | 4.955 | **1.726** |
| AA | 50 | 4.689 | 3.015 | 4.510 | **2.348** | 3.692 | 2.746 |
| | 500 | 5.005 | 3.326 | 4.678 | **1.499** | 4.250 | 2.241 |
| | 5000 | 4.907 | 2.410 | 5.045 | **1.404** | 4.867 | 2.367 |
| AS | 50 | 4.078 | 3.533 | 4.290 | 3.251 | 3.242 | **2.606** |
| | 500 | 5.051 | **2.535** | 4.880 | 2.664 | 3.304 | 2.566 |
| | 5000 | 4.947 | 2.296 | 5.867 | **1.355** | 3.996 | 2.538 |
| Avg. Rank | | 4.779 | 2.813 | 5.028 | 2.336 | 4.016 | **2.027** |
| Drift Type | Size | LS | | | | | |
| SS | 50 | 4.528 | **2.890** | 4.428 | **2.856** | 3.864 | **2.434** |
| | 500 | 5.406 | 3.276 | 4.924 | 2.090 | 3.308 | **1.995** |
| | 5000 | 5.344 | 2.767 | 5.140 | **1.164** | 4.353 | 2.233 |
| IS | 50 | 4.154 | 3.509 | 4.333 | 3.033 | 3.330 | **2.642** |
| | 500 | 5.416 | 2.965 | 5.180 | 2.761 | 3.451 | **1.227** |
| | 5000 | 5.097 | 2.234 | 5.297 | **1.920** | 4.461 | 1.990 |
| II | 50 | 4.538 | 3.654 | 4.581 | 2.964 | 3.136 | **2.128** |
| | 500 | 4.912 | 3.078 | 5.184 | 2.259 | 4.320 | **1.247** |
| | 5000 | 5.360 | 3.031 | 4.539 | 1.692 | 4.955 | **1.424** |
| IA | 50 | 4.757 | 3.360 | 4.927 | 3.207 | 3.262 | **1.486** |
| | 500 | 5.118 | 2.850 | 4.793 | 3.142 | 3.724 | **1.372** |
| | 5000 | 5.280 | 2.378 | 4.760 | 2.179 | 4.795 | **1.608** |
| AA | 50 | 4.436 | 3.288 | 4.190 | **2.386** | 3.663 | 3.037 |
| | 500 | 5.376 | 3.632 | 5.051 | **1.628** | 3.201 | 2.113 |
| | 5000 | 5.151 | 2.476 | 4.364 | **1.035** | 5.301 | 2.673 |
| AS | 50 | 4.143 | 3.809 | 4.195 | **3.134** | 2.917 | **2.802** |
| | 500 | 5.172 | 3.088 | 4.867 | **2.098** | 3.487 | 2.288 |
| | 5000 | 5.646 | 2.736 | 5.008 | **1.139** | 4.037 | 2.433 |
| Avg. Rank | | 4.991 | 3.057 | 4.764 | 2.260 | 3.865 | **2.063** |

evaluation metric are reported in IV. BRPW-MARLENE with similar sources, non-similar sources, and no source achieve the first, second, and third best average ranks across all metrics, respectively. This shows that modelling label dependencies with BRPW-MARLENE consistently outperforms all BR-MARLENE with/without sources. Moreover, BRPW-MARLENE benefits most from similar sources, but non-similar sources can also be helpful.

Among BR-MARLENE methods using different source settings, incorporating similar sources gives the best average performance across all metrics, with no-source BR-MARLENE second best on Macro-G-Mean and Micro-G-Mean, and non-similar sources second on LS-G-Mean. This shows that similar sources offer greater improvement to BR-MARLENE, as also observed for BRPW-MARLENE. It is worth noting that BR/BRPW-MARLENE with similar sources do not always achieve the best ranks (e.g., Macro AA 5000); as the number of target examples increases, both methods can perform well even without source help. Thus, similar sources are most beneficial when the target size is small, which is further investigated in the next section.

To further analyse the performance on each label under different types of concept drift, we plot G-Mean curves against different labels. G-Mean on each label is calculated prequentially over 30 runs and reset at each drift. This isolates performance on each concept. Representative results are shown in Figure 1.

If the concept is stable, similar sources offer the greatest benefit at the beginning of learning (e.g., 1a, 1b, 1d, 1h, 1l) when target examples are limited, but as more target data becomes available, the performance of methods with similar, non-similar, or no source becomes similar. When concept drifts occur frequently, similar sources continue to provide an advantage throughout (e.g., 1c, 1e, 1f, 1h), with greater benefit as drift frequency increases. This is reasonable, as transfer from similar sources is less useful once a large number of target examples is available for training.

### C. Effect Analysis of the Transfer Learning

The strong results of BR/BRPW-MARLENE compared to existing methods (Sections VII-A and VII-B) demonstrate the effectiveness of our approach. However, it remains to be verified whether this improvement is truly due to our designed transfer mechanisms. This section provides an analysis of whether this is the case.

To assess the role of transfer learning in our methods, we select two datasets (AA with similar sources, 5000 samples; and Yeast) and plot the average weight ratios of all source sub-classifiers (i.e., all sub-classifiers trained on source streams or on other labels in the target stream, excluding the given target label) over 30 runs. For BR-MARLENE, the source sub-classifier weight ratio ($SWR^q$) is the percentage of total weights assigned to source sub-classifiers in the ensemble for label $l_{T,q}$. The average source weight ratio ($ASWR$) is then calculated as:

$$SWR^q = \frac{\sum_{h \in \mathcal{H}_{src}^{BR}} \alpha_h^q}{\sum_{h \in \mathcal{H}^{BR}} \alpha_h^q}, \quad ASWR = \frac{\sum_{q=1}^{|\mathcal{L}_T|} SWR^q}{|\mathcal{L}_T|} \quad (22)$$

where $\mathcal{H}_{src}$ is the set of all source sub-classifiers. For BRPW-MARLENE, the average weight ratio of source PW-classifiers
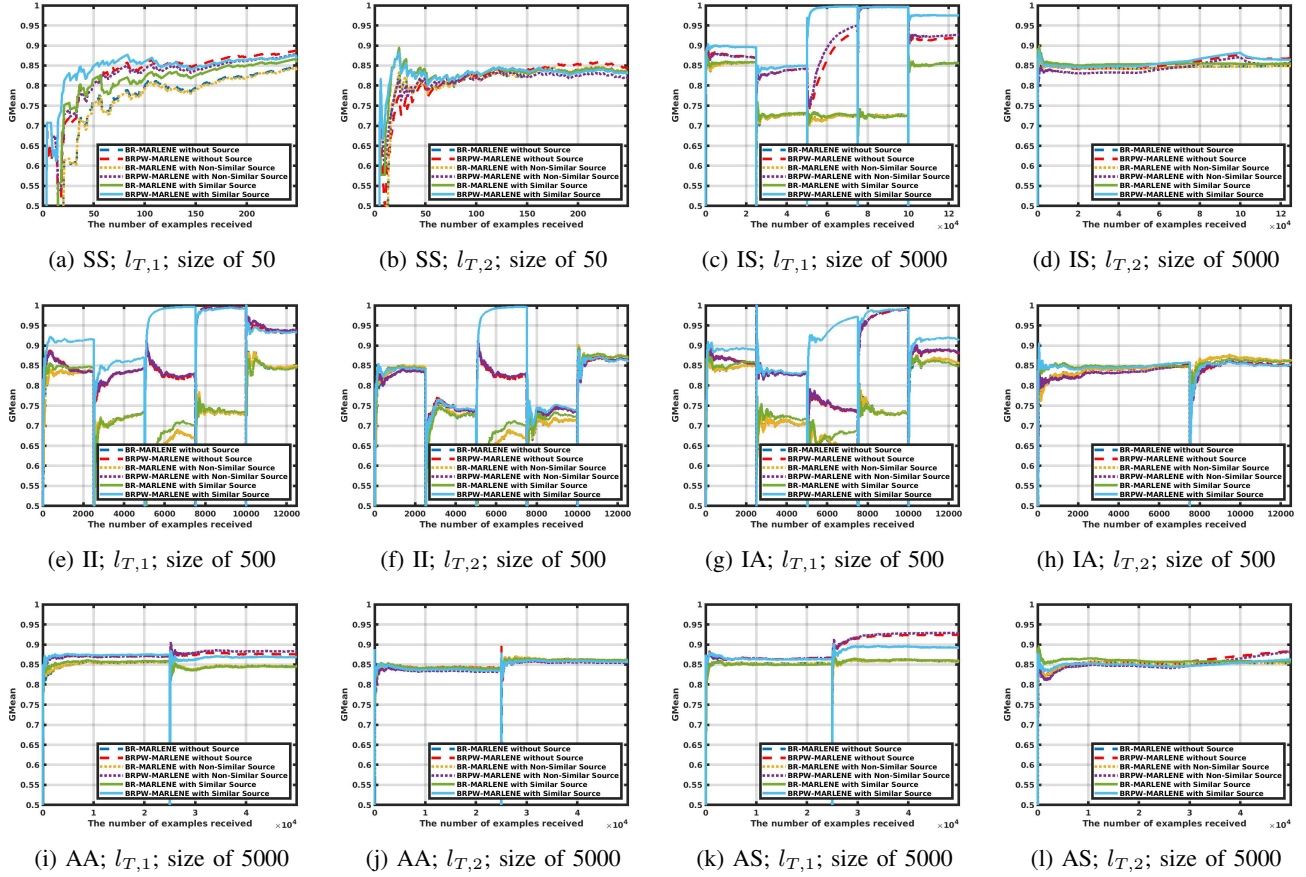
Fig. 1: Average G-Mean of label $l_{T,1}$ and $l_{T,2}$ on Synthetic Datasets.

is calculated as:

$$SWR^{q,q'} = \frac{\sum_{h \in \mathcal{H}_{src}^{PW}} \alpha_h^{q,q'}}{\sum_{h \in \mathcal{H}^{PW}} \alpha_h^{q,q'}} \qquad (23)$$

$$ASWR = \frac{\sum_{q=1}^{|\mathcal{L}_T|} \sum_{q'=1, q' \neq q}^{|\mathcal{L}_T|} SWR^{q,q'}}{|\mathcal{L}_T||\mathcal{L}_T| - 1} \qquad (24)$$

Figure 2 shows that the $ASWR$ of both BR- and BRPW-MARLENE remains high, indicating a strong contribution from source sub-classifiers over time. For the synthetic AA dataset, this weight reflects 1) transfer from source data streams to the target stream and 2) transfer between different labels within the target stream. For Yeast, it reflects inter-label transfer. The consistently high ASWR values confirm that both source-to-target and inter-label transfer are beneficial.

The source PW-classifiers in BRPW-MARLENE have a higher $ASWR$ than source BR-classifiers in BR-MARLENE, since BRPW-MARLENE's ensemble contains many more sub-classifiers; even with some low-weight source sub-classifiers, the total source weight remains high. Spikes and drops in $ASWR$ for both methods on the real-world dataset (Figures 2b, 2d) suggest that the weighting mechanism may be affected by noise.
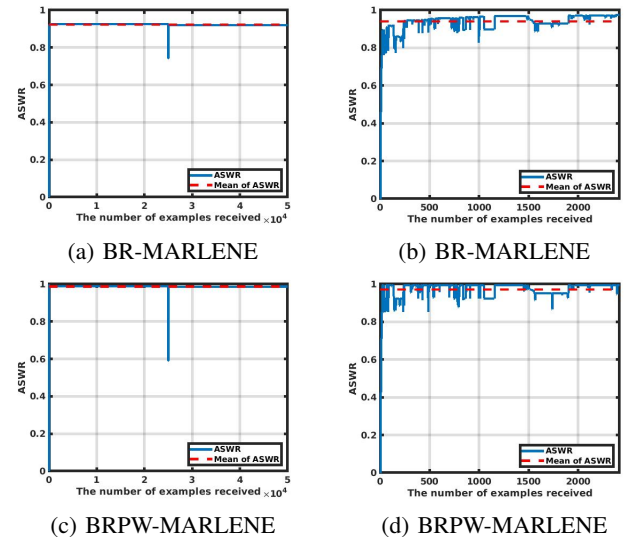


Fig. 2: Average Source sub-classifiers' Weight Ratio. (a) and (c) show results on AA (size 5000, Similar Source); (b) and (d) show results on Yeast.

## VIII. Conclusion

In this paper, we introduced BR-MARLENE, the first approach to transfer knowledge across different labels to boost multi-label classification performance in non-stationary environments. Building on this, we proposed BRPW-MARLENE, a novel extension that further improves classification performance by modelling pairwise label dependencies and transferring knowledge between these dependencies. We conduct comprehensive experiments on real-world and synthetic datasets. The results demonstrate both the effectiveness and clear advantage of these transfer mechanisms.

Although BRPW-MARLENE delivers higher predictive performance, its pairwise dependency modelling is not scalable for problems with a large number of labels. Making BRPW-MARLENE more efficient for large-scale data is a key challenge. Other future directions include experimenting with alternative base classifiers, drift detectors and additional datasets.

## References

[1] J. Read and I. Žliobaitė, "Learning from data streams: An overview and update," *arXiv preprint arXiv:2212.14720*, 2022.

[2] E. Spyromitros-Xioufis, M. Spiliopoulou, G. Tsoumakas, and I. Vlahavas, "Dealing with concept drift and class imbalance in multi-label stream classification," *Department of Computer Science, Aristotle University of Thessaloniki*, 2011.

[3] A. Büyükçakir, H. Bonab, and F. Can, "A novel online stacked ensemble for multi-label stream classification," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1063–1072.

[4] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[5] M. Kubat, S. Matwin *et al.*, "Addressing the curse of imbalanced training sets: one-sided selection," in *14th International Conference on Machine Learning*, vol. 97. Citeseer, 1997, pp. 79–86.

[6] L. L. Minku and X. Yao, "Ddd: A new ensemble approach for dealing with concept drift," *IEEE transactions on knowledge and data engineering*, vol. 24, no. 4, pp. 619–633, 2011.

[7] P. Zhao, S. C. Hoi, J. Wang, and B. Li, "Online transfer learning," *Artificial Intelligence*, vol. 216, pp. 76–102, 2014.

[8] H. Du, L. L. Minku, and H. Zhou, "Multi-source transfer learning for non-stationary environments," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[9] E. Yu, J. Lu, B. Zhang, and G. Zhang, "Online boosting adaptive learning under concept drift for multistream classification," in *AAAI Conference on Artificial Intelligence*, vol. 38, no. 15, 2024, pp. 16 522–16 530.

[10] H. Du, L. L. Minku, and H. Zhou, "Marline: Multi-source mapping transfer learning for non-stationary environments," in *IEEE International Conference on Data Mining*. IEEE, 2020, pp. 122–131.

[11] Y. Sun, K. Tang, Z. Zhu, and X. Yao, "Concept drift adaptation by exploiting historical knowledge," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4822–4832, 2018.

[12] C. Yang, Y.-m. Cheung, J. Ding, and K. C. Tan, "Concept drift-tolerant transfer learning in dynamic environments," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 8, pp. 3857–3871, 2021.

[13] B. Jiao and S. Liu, "Otl-ce: Online transfer learning for data streams with class evolution," *Neurocomputing*, p. 129470, 2025.

[14] X. Renchunzi and M. Pratama, "Automatic online multi-source domain adaptation," *Information Sciences*, vol. 582, pp. 480–494, 2022.

[15] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2004, pp. 22–30.

[16] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2009, pp. 254–269.

[17] J. Fürnkranz, E. Hüllermeier, E. L. Mencía, and K. Brinker, "Multilabel classification via calibrated label ranking," *Machine Learning*, vol. 73, no. 2, pp. 133–153, 2008.

[18] J. Read, B. Pfahringer, and G. Holmes, "Multi-label classification using ensembles of pruned sets," in *2008 eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 995–1000.

[19] A. Osojnik, P. Panov, and S. Džeroski, "Multi-label classification via multi-target regression on data streams," *Machine Learning*, vol. 106, no. 6, pp. 745–770, 2017.

[20] H. Wu, M. Han, Z. Chen, M. Li, and X. Zhang, "A weighted ensemble classification algorithm based on nearest neighbors for multi-label data stream," *ACM Transactions on Knowledge Discovery from Data*, vol. 17, no. 5, pp. 1–21, 2023.

[21] Y. Zou, X. Hu, P. Li, and J. Hu, "Weak multi-label data stream classification under distribution changes in labels," *IEEE Transactions on Big Data*, 2024.

[22] P. Li, H. Zhang, X. Hu, and X. Wu, "High-dimensional multi-label data stream classification with concept drifting detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 8, pp. 8085–8099, 2022.

[23] A. Esteban, A. Cano, A. Zafra, and S. Ventura, "Hoeffding adaptive trees for multi-label classification on data streams," *Knowledge-Based Systems*, vol. 304, p. 112561, 2024.

[24] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2010, pp. 135–150.

[25] J. Read, A. Bifet, G. Holmes, and B. Pfahringer, "Scalable and efficient multi-label classification for evolving data streams," *Machine Learning*, vol. 88, no. 1-2, pp. 243–272, 2012.

[26] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, pp. 71–80.

[27] S. Wang, L. L. Minku, D. Ghezzi, D. Caltabiano, P. Tino, and X. Yao, "Concept drift detection for online class imbalance learning," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2013, pp. 1–10.

[28] D. You, Y. Wang, J. Xiao, Y. Lin, M. Pan, Z. Chen, L. Shen, and X. Wu, "Online multi-label streaming feature selection with label correlation," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[29] R. Sousa and J. Gama, "Multi-label classification from high-speed data streams with adaptive model rules and random rules," *Progress in Artificial Intelligence*, vol. 7, no. 3, pp. 177–187, 2018.

[30] M. Vergara, B. Bustos, I. Sipiran, T. Schreck, and S. Lengauer, "Multi-label learning on low label density sets with few examples," *Expert Systems with Applications*, vol. 265, p. 125942, 2025.

[31] X. Zheng, P. Li, Z. Chu, and X. Hu, "A survey on multi-label data stream classification," *IEEE Access*, vol. 8, pp. 1249–1275, 2019.

[32] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *The Journal of Machine Learning Research*, vol. 11, no. May, pp. 1601–1604, 2010.

[33] J. Read, P. Reutemann, B. Pfahringer, and G. Holmes, "Meka: a multi-label/multi-target extension to weka," in *The Journal of Machine Learning Research*, 2016.

[34] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavalda, "New ensemble methods for evolving data streams," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, 2009, pp. 139–148.

[35] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[36] G. Wu and J. Zhu, "Multi-label classification: do hamming loss and subset accuracy really conflict with each other?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 3130–3140, 2020.

[37] M. Heydarian, T. E. Doyle, and R. Samavi, "Mlcm: Multi-label confusion matrix," *Ieee Access*, vol. 10, pp. 19 083–19 095, 2022.

[38] F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera, "Addressing imbalance in multilabel classification: Measures and random resampling algorithms," *Neurocomputing*, vol. 163, pp. 3–16, 2015.

[39] S. Wang, L. L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1356–1368, 2014.