Imbalanced Data Stream Classification Using Hybrid Data Preprocessing

Barbara Bobowska(✉), Jakub Klikowski(✉), and Michał Woźniak(✉)

Department of Systems and Computer Networks,
Wrocław University of Science and Technology, Wrocław, Poland
{barbara.bobowska,jakub.klikowski,michal.wozniak}@pwr.edu.pl

Abstract. Imbalanced data streams have gained significant popularity among the researchers in recent years. This area of research is not only still greatly underdeveloped, but there are also numerous inherent difficulties that need to be addressed when creating algorithms that could be utilized in such dynamic environment and achieve satisfactory results when it comes to their predictive abilities. In this paper, a novel algorithm that combines both over- and under-sampling techniques in order to create a more robust classifier dedicated to imbalanced data streams is proposed. The efficiency and high predictive quality of the proposed method have been confirmed on the basis of extensive experimental research carried out on the real and the computer-generated data streams.

Keywords: Imbalanced data · Data stream classification · Data preprocessing

1 Introduction

In the last couple of years, a sharp rise in products and systems using machine learning to enhance their performance is observed. Many of the applications such as predicting user behavior on social platforms like Twitter, or client activity on online stores fall into the category of imbalanced data stream classification [24]. When designing methods for data stream classification one has to take into account the characteristics of a data stream such as the sequential manner that the data arrives, over which one has no control when it comes to the order of the arriving samples, as well as the fact that the size of the stream could be possibly infinite. Due to that requirement, it is impossible to process the upcoming data in multiple passes and such the samples can be processed once [26]. Furthermore, one has to consider the rapid rate at which the data arrives, at the same time ensuring that the processing of the data stream is done in a timely fashion such that the delay in the performance of the algorithm is minimal. Data streams can exhibit a change in data and target concepts over time (so-called Non-stationary data streams) [16,26]. Such a phenomenon is called *concept drift* [12] and it is quite common i.e. the change of popular topics on Twitter.

P. Cellier and K. Driessens (Eds.): ECML PKDD 2019 Workshops, CCIS 1168, pp. 402–413, 2020.
https://doi.org/10.1007/978-3-030-43887-6_33

Due to the *concept drift* the performance of the classifier can degrade over time and as such the classifier has to be trained incrementally to accommodate the changes of concepts of non-stationary data streams. Moreover, the proportion between classes is often skewed with one class being over-represented. In cases where the imbalance ratio is present traditional accuracy driven methods are not applicable especially when misclassification of the minority class examples is much more costly, as is often the case i.e. fraud detection [24]. It is worth mentioning that not only the imbalance ratio can influence the performance of the classifier. Some examples can be easy to classify even when the IR is high if the classes are separated from each other the decision boundary and be determined with ease. However, it has been observed that instances of the minority class have a tendency to create sparsely spread throughout the object space clusters, often surrounded by majority class examples [4]. The presence of noise and outliers is another difficulty factor that needs to be addressed. In [3,15] authors created preprocessing methods with those issues in mind.

Data streams may be processed either in blocks or one instance at a time. One of the most important issues in learning from the data stream is when to update the classifier [22]. Most researchers distinguish between two approaches: active and passive. In the former, the update is performed only if drift is detected while the later updates the classifier continuously regardless if the drift was detected or not [9]. In order to satisfy the time and memory requirements, a forgetting or data management mechanism must be used. One of the most popular approaches to forgetting is using sliding windows, which can be either sequence based, where the size of the window is defined by a number of instances and time stamp based where the window is defined by a certain duration time. In the simplest example sliding windows are of fixed size, and include only the most recent examples. The oldest samples in a window are discarded in favor of new ones. Some methods implement sliding windows of varying size depending on the response from drift detectors [2].

The main contributions of this work are as follows:

- Proposition of the two novel imbalanced data stream classifiers (DSC-R and DSC-S) which employ under- and oversampling techniques for balancing data.
- Experimental evaluation of the proposed algorithms and their comparison with state-of-art methods.

The article is organized as follows. Sections 1 and 2 present a brief introduction to the problem of imbalance data stream classification and a quick overview of the state-of-the-art algorithms dedicated to it. Section 3 offers an in-depth explanation of the proposed solution. Section 4 showcases the results of the computer experiments, comparing the proposed algorithm to different techniques for imbalanced data classification, proving the usefulness of the developed algorithm. Section 5 presents the conclusions and describes possible future improvements to the proposed method.

2 Related Works

Studies over the years presented algorithms dedicated to data stream analysis. Very fast decision tree (VFDT) proposed by Domingos and Hulten [13] was among the first methods for stream analysis, that to this day has been a basis for many modifications. VFDT utilizes the Hoeffding bound in order to calculate the proper number of examples needed to select the split-node. The algorithm incrementally creates a tree form from a data stream ensuring that once the examples were used to update the tree they are negligible and can be removed. The aforementioned modifications include ideas such as pruning mechanisms or utilizing sliding windows and drift detectors in order to better the algorithms in case of non-stationary streams [10]. Worth noting are several methods using ensembles of classifiers. Weighted Majority Algorithm [18] adjusts the weights of the classifiers in the ensemble so that the weight of an expert that misclassified an instance is decreased accordingly to the user-specified value. A modification of the method with an added procedure which adds new classifiers to the ensemble when the overall performance is unsatisfactory called Dynamic Weighted Majority (DWM) was introduced in [14]. In Accuracy Weighted Ensemble (AWE) a new classifier is added only if the ensemble's size is not exceeded [25] while in Learn++.NSE [8] such a constraint is not applied. In Learn++.CDS Ditzler and Polikar combine their previous work Learn++.NSE with SMOTE sampling in order to better address the data imbalance and later replacing SMOTE with an original bagging-based method of data balancing [7]. In SEA [23] a new classifier candidate is evaluated to determine whether or not it is worth including into the ensemble at the cost of replacing some other classifier already in the ensemble. Other approaches such as OUSEnsemble (Over Under Sampling Ensemble) [11] make use of sampling techniques. The stream is divided into blocks that consist of examples from both majority and minority class. The idea is to propagate all the instances of the minority class from the previous block and under-sample the majority examples in the current block such that the desired imbalance ratio is acquired. Afterwards, from the resultant subset, datasets later used to build component classifiers for the ensemble, are created by propagating all instances of the minority class to each of the datasets while each example from the majority class is propagated to only one dataset. Proposed by Chen and He the Selectively Recursive Approach (SERA) [5] uses a Mahalanobis distance to determine which of the examples from the minority class in the previous block are most similar to the minority examples in the current block. Based on that a limited number of minority class examples is selected and added to the majority class examples in the current block. Chen and He later designed a Recursive Ensemble Approach (REA) [6]. In REA minority class examples from the previous block that are nearest neighbors of minority class examples from the current block are added in order to balance the given training block. Both REA and SERA proved to make more accurate predictions than the method proposed by [19]. A Chunk-based ensemble approach, proposed by Wang et al. called KMean-Clustering [25] utilizes k-mean clustering in order to under-sample the majority class, by using the centroids created in the clustering process to resample the majority instances.

3 The Deterministic Sampling Classifier

The proposed method, called *Deterministic Sampling Classifier* (DSC), for data stream classification, processes the upcoming data in chunks. Each chunk is used in two operations. Firstly, the instances of the majority class present in the currently processed block are under-sampled in order to produce a balanced class representation in a data chunk (Fig. 1).

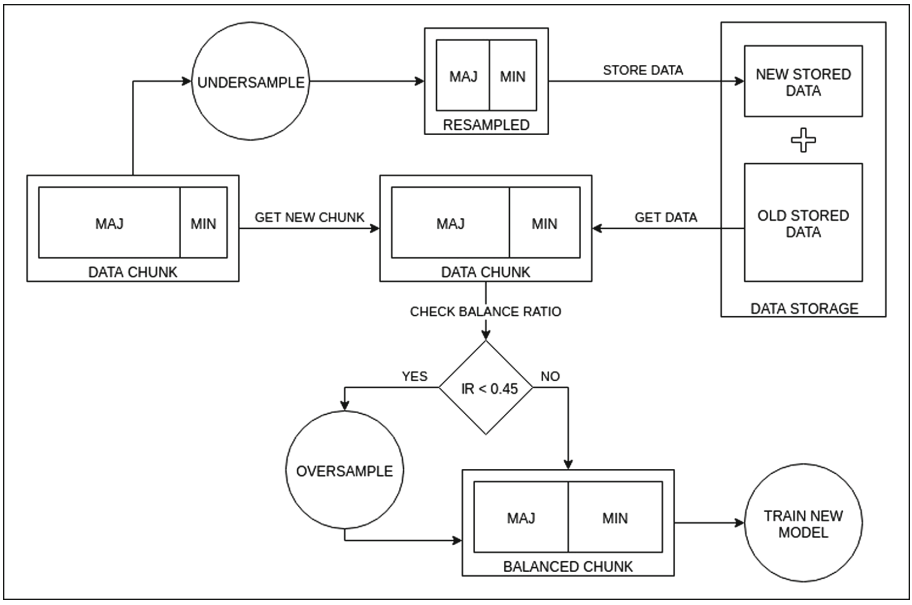

Fig. 1. Proposed method flow diagram

The resulting data (referred to in the Fig. 1 as NEW STORED DATA) is then stored in a memory buffer (DATA STORAGE). Secondly, that same block of data is combined with a part of the data from the buffer, called OLD STORED DATA, using GET NEW CHUNK, which copies the data from the currently processed block and the GET DATA method, which takes OLD STORED DATA from the DATA STORAGE buffer. OLD STORED DATA, consists of all the previously accumulated under-sampled chunks. When a new chunk of data arrives the data from NEW STORED DATA are moved to the OLD STORED DATA part of the buffer. The DATA STORAGE is of fixed size. When the buffer is full, the oldest examples are removed from it. Afterward, the imbalance ratio of the data block created as a result of the GET NEW CHUNK and GET DATA is calculated, and if the value is lower than 0.45, an oversampling of the minority class is performed, and then a classification model is trained. Otherwise, the algorithm accepts the chunk as properly balanced and uses it to train the

model right away. The implementation allows one to choose sampling algorithms of their liking. In this paper, the authors created two versions of the method DSC-S (Deterministic Sampling Classifier-SMOTE) and DSC-R (Deterministic Sampling Classifier-Random). For the DSC-R method the chosen sampling methods were: random over- and under-sampling and for the DSC-S: SMOTE and NearMiss (implementation from the imbalanced-learn library [17]) for over- and under-sampling accordingly.

4 Experimental Evaluation

The quality of the proposed algorithms was evaluated on the basis of computer experiments, using 26 real and 60 synthetic data streams. The evaluation procedure used in order to assess the predictive performance of a data stream classifier was conducted by interleaving testing with training (test-then-train) [16]. Each block is first used to test the classifier and afterward it is used for training. As a measure of comparison, the following methods were used: OUSEnsemble, KMeanClustering, REA, Learn++.CDS, Learn++.NIE and MLPClassifier (Multi-layer Perceptron classifier), using a k-NN as a base classifier. The algorithms were implemented in Python using Scikit-learn [20] and imbalanced-learn [17] libraries[1]. The selected real streams were downloaded from the KEEL [1] and PROMISE Software Engineering Repository [21]. The chosen datasets consisted of multidimensional binary classification problems with the imbalance ratio ranging from 1 to 39. The datasets were described in Table 1. The results were analyzed using the KEEL software evaluation tool [1]. Non-parametrical statistical tests were performed namely the Friedman Test as well as a Nemenyi's Post-Hoc Procedure. The metrics chosen were F-score, Gmean and AUC score. Tables 2, 3 and 4 present the obtained results. The table presents the obtained results as the mean value for each of the metrics, as well as, the information on those methods that performed poorly in comparison with the method named in the column, placed directly below the score. For instance, given the abalone-17-vs-7-8-9-10 dataset, the DSC-R algorithm performed statistically better than the 3rd, 5th, the 7th and 8th algorithm in the table (read from left to right). The obtained results prove the usefulness of the proposed algorithms. For the F-score the proposed DSC-R and DSC-S algorithms along with the REA algorithm have the best results. What is interesting the MLPC algorithm performed consistently the worst. For the Gmean the results are similar. The methods introduced in the paper perform favorably in comparison with other algorithms, the Learn++.CDS and Learn++.NIE techniques, as well as REA, have comparable results to the DSC-R and DSC-S methods. Lastly, the results in Table 4 representing the results for AUC score indicate the proposed algorithm obtained satisfactory results,

[1] Repository link: https://github.com/w4k2/iot-ecml2019.

with only the LCDS algorithm performing marginally better. It is worth mentioning, that the created methods are robust enough, so that imbalance ratio (whether low or high) does not negatively impact their performance.

Table 1. Overview of datasets selected for experimental evaluation (source: KEEL and PROMISE Software Engineering Repository).

Dataset	IMB. RATIO	SAMPLES	FEATURES
abalone-17_vs_7-8-9-10	39	2338	8
australian	1	690	14
glass-0-1-2-3-vs-4-5-6	3	214	9
glass0	2	214	9
glass1	2	214	9
heart	1	270	13
jm1	4	10885	22
kc1	5	2109	22
kc2	4	522	22
kr-vs-k-three_vs_eleven	35	2935	6
kr-vs-k-zero-one_vs_draw	26	2901	6
page-blocks0	9	5472	10
pima	2	768	8
segment0	6	2308	19
shuttle-c0-vs-c4	14	1829	9
vehicle0	3	846	18
vowel0	10	988	13
wisconsin	2	683	9
yeast-0-2-5-6-vs-3-7-8-9	9	1004	8
yeast-0-2-5-7-9-vs-3-6-8	9	1004	8
yeast-0-3-5-9-vs-7-8	9	506	8
yeast-0-5-6-7-9-vs-4	9	528	8
yeast-2-vs-4	9	514	8
yeast1	2	1484	8
yeast3	8	1484	8

Table 2. Overview of the results for F-score.

Dataset	1	2	3	4	5	6	7	8
	DSC-R	DSC-S	KMC	LCDS	LNIE	REA	OUSE	MLPC
abalone-17-vs-7-8-9-10	0.237	0.208	0.060	0.250	0.046	0.176	0.072	0.000
	3,5,7,8	3,5,7,8	8	3,5,7,8	–	–	8	–
australian	0.571	0.601	0.444	0.514	0.604	0.518	0.602	0.000
	3,8	3,6,8	8	8	3,4,6,8	8	3,4,6,8	–
electricity-normalized	0.631	0.623	0.580	0.588	0.598	0.575	0.593	0.324
	3,4,5,6,7,8	3,4,6,7,8	8	8	8	8	8	–
glass-0-1-2-3-vs-4-5-6	0.765	0.801	0.781	0.814	0.768	0.782	0.452	0.000
	7,8	7,8	7,8	7,8	7,8	7,8	8	–
glass0	0.662	0.673	0.592	0.640	0.516	0.597	0.473	0.000
	5,7,8	5,7,8	7,8	7,8	8	8	8	–
glass1	0.687	0.649	0.576	0.517	0.558	0.412	0.573	0.000
	4,5,6,8	6,8	6,8	8	6,8	8	6,8	–
heart	0.583	0.569	0.489	0.569	0.641	0.538	0.580	0.144
	8	8	8	8	3,8	8	8	–
jm1	0.395	0.390	0.182	0.399	0.372	0.331	0.317	0.170
	3,7,8	3,7,8	–	3,6,7,8	3,8	3,8	3,8	–
kc1	0.387	0.394	0.186	0.377	0.394	0.331	0.311	0.127
	3,7,8	3,6,7,8	–	3,7,8	3,7,8	3,8	3,8	–
kc2	0.532	0.560	0.198	0.511	0.509	0.467	0.380	0.389
	3	3,7,8	–	3	3	3	3	3
kr-vs-k-three-vs-eleven	0.701	0.847	0.256	0.730	0.408	0.836	0.335	0.000
	3,5,7,8	1,3,4,5,7,8	8	3,5,7,8	3,8	1,3,4,5,7,8	8	–
kr-vs-k-zero-one-vs-draw	0.673	0.804	0.395	0.773	0.591	0.785	0.367	0.000
	3,7,8	1,3,5,7,8	8	1,3,5,7,8	3,7,8	1,3,5,7,8	8	–
page-blocks0	0.541	0.576	0.252	0.359	0.335	0.533	0.352	0.323
	3	3,4,5,7,8	–	–	–	3,7	–	–
pima	0.601	0.578	0.508	0.539	0.536	0.581	0.518	0.370
	3,5,7,8	8	8	8	8	8	8	–
segment0	0.671	0.792	0.405	0.587	0.339	0.580	0.294	0.123
	3,4,5,6,7,8	1,3,4,5,6,7,8	7,8	3,5,7,8	8	3,5,7,8	8	–
shuttle-c0-vs-c4	0.995	0.995	0.923	0.960	0.931	0.995	0.955	0.202
	3,5,7,8	3,5,7,8	8	8	8	3,5,7,8	8	–
vehicle0	0.812	0.824	0.653	0.744	0.563	0.758	0.563	0.257
	3,5,7,8	3,4,5,7,8	8	5,7,8	8	5,7,8	8	–
vowel0	0.682	0.753	0.274	0.553	0.272	0.553	0.251	0.059
	3,5,7,8	3,4,5,6,7,8	8	3,5,7,8	8	3,5,7,8	8	–
wisconsin	0.966	0.951	0.708	0.942	0.941	0.949	0.862	0.019
	3,8	3,8	8	3,8	3,8	3,8	3,8	–
yeast-0-2-5-6-vs-3-7-8-9	0.446	0.393	0.358	0.362	0.389	0.514	0.257	0.073
	7,8	8	8	8	8	3,4,7,8	8	–
yeast-0-2-5-7-9-vs-3-6-8	0.667	0.710	0.580	0.675	0.590	0.722	0.366	0.000
	7,8	3,5,7,8	7,8	7,8	7,8	3,5,7,8	8	–
yeast-0-3-5-9-vs-7-8	0.251	0.288	0.222	0.265	0.145	0.403	0.185	0.018
	8	8	8	8	–	3,5,7,8	8	–
yeast-0-5-6-7-9-vs-4	0.350	0.432	0.327	0.428	0.257	0.379	0.176	0.000
	7,8	7,8	7,8	7,8	8	7,8	8	–
yeast-2-vs-4	0.615	0.670	0.581	0.569	0.571	0.702	0.389	0.000
	7,8	7,8	8	8	8	7,8	8	–
yeast1	0.527	0.495	0.520	0.504	0.430	0.476	0.498	0.000
	8	8	8	8	8	8	8	–
yeast3	0.566	0.589	0.429	0.573	0.436	0.644	0.314	0.100
	7,8	7,8	8	7,8	8	7,8	8	–

Table 3. Overview of the results for Gmean score.

Dataset	1	2	3	4	5	6	7	8
	DSC-R	DSC-S	KMC	LCDS	LNIE	REA	OUSE	MLPC
abalone-17-vs-7-8-9-10	0.782	0.818	0.541	0.684	0.102	0.282	0.536	0.000
	3,5,6,7,8	3,5,6,7,8	5,8	5,6,8	–	–	5,8	–
australian	0.622	0.640	0.433	0.557	0.187	0.580	0.100	0.000
	3,5,7,8	3,4,5,6,7,8	5,7,8	3,5,7,8	8	3,5,7,8	–	–
electricity-normalized	0.682	0.676	0.639	0.644	0.510	0.635	0.284	0.393
	3,4,5,6,7,8	3,4,5,6,7,8	5,7,8	5,7,8	7,8	5,7,8	–	7
glass-0-1-2-3-vs-4-5-6	0.865	0.892	0.836	0.905	0.869	0.869	0.165	0.000
	7,8	7,8	7,8	7,8	7,8	7,8	–	–
glass0	0.739	0.756	0.588	0.720	0.287	0.697	0.071	0.000
	3,5,7,8	3,5,7,8	5,7,8	5,7,8	8	5,7,8	–	–
glass1	0.750	0.717	0.560	0.600	0.254	0.526	0.127	0.000
	3,4,5,6,7,8	3,4,5,6,7,8	5,7,8	5,7,8	8	5,7,8	–	–
heart	0.623	0.618	0.518	0.589	0.346	0.566	0.284	0.205
	3,8	3,8	8	8	–	8	–	–
jm1	0.660	0.666	0.275	0.675	0.630	0.557	0.369	0.291
	3,6,7,8	3,6,7,8	–	3,6,7,8	3,7,8	3,7,8	–	–
kc1	0.651	0.670	0.278	0.655	0.658	0.557	0.391	0.204
	3,6,7,8	3,6,7,8	–	3,6,7,8	3,6,7,8	3,7,8	8	–
kc2	0.717	0.772	0.230	0.711	0.715	0.618	0.259	0.347
	3,7,8	3,7,8	–	3,7,8	3,7,8	3,7	–	–
kr-vs-k-three-vs-eleven	0.988	0.985	0.911	0.981	0.836	0.985	0.933	0.000
	3,7,8	3,7,8	8	3,7,8	8	3,7,8	8	–
kr-vs-k-zero-one-vs-draw	0.970	0.965	0.898	0.932	0.873	0.933	0.935	0.000
	3,5,7,8	3,5,8	8	8	8	8	8	–
page-blocks0	0.850	0.848	0.543	0.586	0.649	0.812	0.805	0.447
	3,4,5,8	3,4,5,8	–	–	–	3,8	3,8	–
pima	0.690	0.673	0.590	0.634	0.400	0.670	0.197	0.482
	3,4,5,7,8	3,5,7,8	5,7,8	5,7,8	7	3,5,7,8	–	7
segment0	0.911	0.934	0.719	0.848	0.585	0.851	0.415	0.242
	3,4,5,6,7,8	3,4,5,6,7,8	5,7,8	3,5,7,8	7,8	3,5,7,8	8	–
shuttle-c0-vs-c4	0.995	0.995	0.949	0.963	0.950	0.995	0.996	0.304
	8	8	8	8	8	8	3,5,8	–
vehicle0	0.916	0.901	0.819	0.881	0.723	0.882	0.708	0.380
	3,5,7,8	3,5,7,8	5,7,8	5,7,8	8	5,7,8	–	–
vowel0	0.937	0.938	0.523	0.797	0.634	0.790	0.634	0.224
	3,4,5,7,8	3,4,5,7,8	8	3,5,7,8	8	3,8	8	–
wisconsin	0.974	0.960	0.747	0.953	0.950	0.957	0.781	0.042
	3,8	3,8	8	3,8	3,8	3,8	8	–
yeast-0-2-5-6-vs-3-7-8-9	0.761	0.661	0.685	0.709	0.683	0.745	0.511	0.178
	7,8	8	7,8	7,8	8	7,8	8	–
yeast-0-2-5-7-9-vs-3-6-8	0.862	0.848	0.827	0.879	0.839	0.875	0.681	0.000
	7,8	7,8	7,8	7,8	7,8	7,8	8	–
yeast-0-3-5-9-vs-7-8	0.617	0.650	0.517	0.596	0.339	0.651	0.470	0.069
	8	7,8	8	8	–	8	8	–
yeast-0-5-6-7-9-vs-4	0.750	0.822	0.777	0.783	0.612	0.694	0.485	0.000
	7,8	7,8	7,8	7,8	8	8	8	–
yeast-2-vs-4	0.827	0.826	0.801	0.830	0.723	0.847	0.731	0.000
	8	8	8	8	8	8	8	–
yeast1	0.634	0.616	0.599	0.612	0.330	0.593	0.344	0.000
	5,7,8	5,7,8	5,7,8	5,7,8	8	5,7,8	8	–
yeast3	0.871	0.855	0.815	0.841	0.773	0.875	0.698	0.197
	7,8	7,8	8	7,8	8	5,7,8	8	–

Table 4. Overview of the results for AUC score.

Dataset	1	2	3	4	5	6	7	8
	DSC-R	DSC-S	KMC	LCDS	LNIE	REA	OUSE	MLPC
abalone-17-vs-7-8-9-10	0.997	0.989	0.684	0.999	0.828	0.967	0.789	0.329
	3,5,6,7,8	3,5,6,7,8	8	3,5,6,7,8	3,8	3,5,7,8	3,8	–
australian	0.827	0.829	0.551	0.819	0.662	0.803	0.578	0.448
	3,5,7,8	3,5,7,8	8	3,5,7,8	3,7,8	3,5,7,8	8	–
electricity-normalized	0.961	0.961	0.927	0.974	0.871	0.971	0.551	0.671
	3,5,7,8	3,5,7,8	5,7,8	1,2,3,5,6,7,8	7,8	1,2,3,5,7,8	–	7
glass-0-1-2-3-vs-4-5-6	0.988	0.989	0.894	0.989	0.879	0.984	0.731	0.722
	7,8	7,8	8	7,8	8	7,8	–	–
glass0	0.937	0.929	0.792	0.930	0.724	0.881	0.628	0.675
	3,5,7,8	3,5,7,8	5,7	3,5,7,8	–	5,7,8	–	–
glass1	0.913	0.918	0.681	0.830	0.668	0.790	0.546	0.669
	3,5,6,7,8	3,5,6,7,8	–	5,7	–	7	–	–
heart	0.846	0.855	0.728	0.877	0.662	0.860	0.632	0.374
	3,5,7,8	3,5,7,8	8	3,5,7,8	8	3,5,7,8	8	–
jm1	0.866	0.862	0.398	0.906	0.788	0.858	0.720	0.467
	3,5,7,8	3,5,7,8	–	1,2,3,5,6,7,8	3,7,8	3,5,7,8	3,8	–
kc1	0.865	0.860	0.405	0.902	0.786	0.858	0.734	0.399
	3,5,7,8	3,5,7,8	–	1,2,3,5,6,7,8	3,7,8	3,5,7,8	3,8	–
kc2	0.877	0.889	0.487	0.908	0.832	0.882	0.769	0.757
	3,8	3,7,8	–	3,7,8	3	3,7,8	3	3
kr-vs-k-three-vs-eleven	1.000	1.000	0.981	1.000	0.987	1.000	0.978	0.968
	3,5,7,8	3,5,7,8	–	3,5,7,8	7,8	3,5,7,8	–	–
kr-vs-k-zero-one-vs-draw	1.000	1.000	0.972	1.000	0.982	0.999	0.981	0.691
	3,5,7,8	3,5,6,7,8	8	3,5,6,7,8	8	3,5,7,8	8	–
page-blocks0	0.994	0.994	0.850	0.999	0.962	0.992	0.913	0.734
	3,5,7,8	3,5,7,8	–	1,2,3,5,6,7,8	3,7,8	3,5,7,8	–	–
pima	0.870	0.871	0.736	0.890	0.725	0.850	0.645	0.508
	3,5,7,8	3,5,7,8	7,8	3,5,7,8	7,8	3,5,7,8	8	–
segment0	0.996	0.997	0.859	0.993	0.831	0.975	0.794	0.628
	3,5,6,7,8	3,4,5,6,7,8	7,8	3,5,6,7,8	7,8	3,5,7,8	8	–
shuttle-c0-vs-c4	1.000	1.000	0.999	1.000	0.500	1.000	0.999	0.480
	5,8	5,8	5,8	5,8	–	5,8	5,8	–
vehicle0	0.988	0.990	0.881	0.987	0.834	0.970	0.841	0.767
	3,5,6,7,8	3,5,6,7,8	8	3,5,7,8	–	3,5,7,8	–	–
vowel0	0.997	0.999	0.852	0.999	0.862	0.989	0.870	0.260
	3,5,6,7,8	3,5,6,7,8	8	3,5,6,7,8	8	3,5,7,8	8	–
wisconsin	0.998	0.997	0.929	0.997	0.716	0.995	0.958	0.081
	7,8	7,8	8	7,8	8	7,8	8	–
yeast-0-2-5-6-vs-3-7-8-9	0.951	0.943	0.789	0.989	0.807	0.929	0.803	0.444
	3,5,7,8	3,5,7,8	8	1,2,3,5,6,7,8	8	3,5,7,8	8	–
yeast-0-2-5-7-9-vs-3-6-8	0.983	0.979	0.896	0.997	0.925	0.978	0.900	0.252
	3,5,7,8	3,5,7,8	8	2,3,5,6,7,8	8	3,5,7,8	8	–
yeast-0-3-5-9-vs-7-8	0.947	0.940	0.659	0.989	0.741	0.889	0.730	0.270
	3,5,7,8	3,5,7,8	8	3,5,6,7,8	8	3,5,7,8	8	–
yeast-0-5-6-7-9-vs-4	0.984	0.975	0.867	0.994	0.891	0.965	0.828	0.089
	3,5,7,8	3,5,7,8	8	3,5,6,7,8	8	3,5,7,8	8	–
yeast-2-vs-4	0.987	0.986	0.906	1.000	0.911	0.982	0.891	0.326
	3,7,8	3,7,8	8	1,2,3,5,6,7,8	8	3,7,8	8	–
yeast1	0.887	0.885	0.768	0.922	0.754	0.877	0.655	0.339
	3,5,7,8	3,5,7,8	7,8	3,5,7,8	7,8	3,5,7,8	8	–
yeast3	0.987	0.984	0.919	0.996	0.953	0.982	0.904	0.380
	3,5,7,8	3,5,7,8	8	1,2,3,5,6,7,8	7,8	3,5,7,8	8	–

5 Conclusions and Future Directions

The proposed in this paper methods for imbalanced stream classification DSC-R and DSC-S performed favorably in comparison with other dedicated algorithms. The evaluation of the predictive abilities of the techniques was conducted on the basis of computer experiments. The obtained results were analyzed using statistical tests and for all the chosen metrics F-score, Gmean and AUC score, the proposed methods obtained satisfactory results, comparable to algorithms such as REA or Learn++.CDS or Learn++.NIE. The algorithm utilizes memory buffer in order to propagate the instances from the previous block that were chosen as the representatives. Since the buffer is of fixed size, after it is full some instances must be removed from it. In the current implementation, the oldest examples are deleted. A more advanced "forgetting" mechanism, that could favor the instances from the minority class and only the instances from the majority that are the best representatives could be introduced in order to further improve the performance of the classifier. Additionally testing other sampling methods for under- and over-sampling may prove to produce better results.

Acknowledgement. This work was supported by the Polish National Science Centre under the grant No. 2017/27/B/ST6/01325 as well as by the statutory funds of the Department of Systems and Computer Networks, Faculty of Electronics, Wroclaw University of Science and Technology.

References

1. Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S.: Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. Mult. Valued Log. Soft Comput. **17**(2–3), 255–287 (2011). http://dblp.uni-trier.de/db/journals/mvl/mvl17.html
2. Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing. In: In SIAM International Conference on Data Mining (2007)
3. Bobowska, B., Woźniak, M.: Experimental study on modified radial-based oversampling. In: Graña, M., et al. (eds.) SOCO'18-CISIS'18-ICEUTE'18 2018. AISC, vol. 771, pp. 110–119. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-94120-2_11
4. Brzezinski, D., Stefanowski, J.: Ensemble classifiers for imbalanced and evolving data streams, pp. 44–68, March 2018. https://doi.org/10.1142/9789813228047_0003
5. Chen, S., He, H.: SERA: selectively recursive approach towards nonstationary imbalanced stream data mining. In: 2009 International Joint Conference on Neural Networks, pp. 522–529, June 2009. https://doi.org/10.1109/IJCNN.2009.5178874
6. Chen, S., He, H.: Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach. Evol. Syst. **2**(1), 35–50 (2011). https://doi.org/10.1007/s12530-010-9021-y
7. Ditzler, G., Polikar, R.: Incremental learning of concept drift from streaming imbalanced data. IEEE Trans. Knowl. Data Eng. **25**(10), 2283–2301 (2013)

8. Ditzler, G., Roveri, M., Alippi, C., Polikar, R.: Learning in nonstationary environments: a survey. IEEE Comput. Intell. Mag. **10**, 12–25 (2015). https://doi.org/10.1109/MCI.2015.2471196
9. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. **46**(4), 44:1–44:37 (2014). https://doi.org/10.1145/2523813. http://doi.acm.org/10.1145/2523813
10. Gama, J.: Knowledge Discovery from Data Streams, 1st edn. Chapman & Hall/CRC, Boca Raton (2010)
11. Gao, J., Ding, B., Fan, W., Han, J., Philip, S.Y.: Classifying data streams with skewed class distributions and concept drifts. IEEE Internet Comput. **12**(6), 37–49 (2008)
12. Hoens, T.R., Polikar, R., Chawla, N.V.: Learning from streaming data with concept drift and imbalance: an overview. Prog. Artif. Intell. **1**(1), 89–101 (2012). https://doi.org/10.1007/s13748-011-0008-0
13. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2001, pp. 97–106. ACM, New York (2001). https://doi.org/10.1145/502512.502529. http://doi.acm.org/10.1145/502512.502529
14. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: a new ensemble method for tracking concept drift. In: Proceedings of the Third IEEE International Conference on Data Mining, ICDM 2003, p. 123. IEEE Computer Society, Washington, D.C. (2003). http://dl.acm.org/citation.cfm?id=951949.952136
15. Koziarski, M., Krawczyk, B., Woźniak, M.: Radial-based approach to imbalanced data oversampling. In: Martínez de Pisón, F.J., Urraca, R., Quintián, H., Corchado, E. (eds.) HAIS 2017. LNCS (LNAI), vol. 10334, pp. 318–327. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59650-1_27
16. Krawczyk, B., Minku, L.L., Gama, J., Stefanowski, J., Woniak, M.: Ensemble learning for data stream analysis. Inf. Fusion **37**(C), 132–156 (2017). https://doi.org/10.1016/j.inffus.2017.02.004
17. Lemaître, G., Nogueira, F., Aridas, C.K.: Imbalanced-learn: a Python toolbox to tackle the curse of imbalanced datasets in machine learning. J. Mach. Learn. Res. **18**(17), 1–5 (2017). http://jmlr.org/papers/v18/16-365.html
18. Littlestone, N., Warmuth, M.K.: The weighted majority algorithm. Inf. Comput. **108**(2), 212–261 (1994). https://doi.org/10.1006/inco.1994.1009
19. Masud, M., Gao, J., Khan, L., Han, J., Thuraisingham, B.: A practical approach to classify evolving data streams: training with limited amount of labeled data, pp. 929–934, December 2008. https://doi.org/10.1109/ICDM.2008.152
20. Pedregosa, F., et al.: Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011). http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf
21. Sayyad Shirabad, J., Menzies, T.: The PROMISE repository of software engineering databases. School of Information Technology and Engineering, University of Ottawa, Canada (2005). http://promise.site.uottawa.ca/SERepository
22. Stefanowski, J., Brzezinski, D.: Stream classification. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning and Data Mining, pp. 1191–1199. Springer, Boston (2017). https://doi.org/10.1007/978-1-4899-7687-1_908

23. Street, W.N., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2001, pp. 377–382. ACM, New York (2001). https://doi.org/10.1145/502512.502568. http://doi.acm.org/10.1145/502512.502568
24. Sun, Y., Wong, A.K.C., Kamel, M.S.: Classification of imbalanced data: a review. IJPRAI **23**, 687–719 (2009)
25. Wang, Y., Zhang, Y., Wang, Y.: Mining data streams with skewed distribution by static classifier ensemble. In: Chien, B.C., Hong, T.P. (eds.) Opportunities and Challenges for Next-Generation Applied Intelligence. SCI, vol. 214, pp. 65–71. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-92814-0_11
26. Woźniak, M., Graña, M., Corchado, E.: A survey of multiple classifier systems as hybrid systems. Inf. Fusion **16**, 3–17 (2014)