Imbalanced data classification using MapReduce and relief

Joanna Jedrzejowicz, Robert Kostrzewski, Jakub Neumann and Magdalena Zakrzewska

Institute of Informatics, Faculty of Mathematics, Physics and Informatics, University of Gdansk, Gdansk, Poland

ABSTRACT

Classification of imbalanced data has been reported to require modification of standard classification algorithms and lately has attracted a lot of attention due to practical applications in industry, banking and finance. The aim of the paper is to examine algorithms known from literature when two modifications are introduced: MapReduce to parallelize computations and Relief to select most valuable attributes. Both modifications are needed in Big Data area. Also two new algorithms are considered.

ARTICLE HISTORY

Received 29 November 2017
Accepted 11 February 2018

KEYWORDS

Imbalanced data; classification; parallelization; feature selection

1. Introduction

The imbalance classification algorithms deal with datasets in which the cardinality of one of the classes is much bigger than that of other classes. In case of binary classification, which is considered in this paper, it means that we deal with data containing instances from two classes and instances from one class, majority class, outnumber instances from the other class, that is minority class.

As follows from many research projects special approach to classification of imbalanced data has to be applied since traditional classification algorithms do not give satisfactory results. There are several reasons why classifying imbalanced data is a challenge. Standard classifiers such as Support Vector Machines, logistic regression, decision trees and Naive Bayes are not suitable because they often give good coverage of the majority examples but minority classes are misrepresented. Even if the obtained accuracy is for example 99% which would be quite satisfactory for usual classification can be misleading if the minority class is 1% of the whole dataset. Minority class examples can be treated as noise by the classifier and, vice versa, noise wrongly identified as minority class. Therefore, special methods are needed.

What is more, it appears that imbalanced data occur often in every day life situations. In Haixiang et al. (2017) they are denoted as rare events and include software defects, natural disasters, fraud detection, text classification, etc. Lately special attention is drawn to algorithms adjusted for application in specific areas, mainly because of competing business issues. In Zhu, Baesens, and vanden Broucke (2017) the area of considered data sets is
customer churn prediction which is important in customer relationship management. Misclassifying mentioned rare events can result in high costs which is another reason for looking for better classification methods for imbalanced data. Recently, a study on imbalanced enterprise credit evaluation appeared in Sun, Lang, Fujita, and Li (2018). It gives a broad report on experiments with data from sample Chinese companies with high credit risk and negative net profit in two considered successive years. Again, it reports on very practical issues and gives motivation to look for more effective and accurate methods for classifying imbalanced data.

This paper is an extended version of the INISTA’2017 conference paper (Jedrzejowicz, Neumann, Synowczyk, and Zakrzewska, 2017) where the MapReduce paradigm was applied to known literature classification algorithms to reduce time processing. Here, the survey is extended by considering the impact of attributes reduction on accuracy (see Table 4), suggesting new Algorithm 4 and modifying algorithm EDBC via introducing other clustering algorithms instead of originally used time-consuming Jarvis Patrick (Zhang et al., 2015). Also, the number of considered datasets was extended, as well as survey comparing results known from the literature was added.

The paper is organized as follows. Section 2 contains a brief overview of recent results in the field. Section 3 presents the general idea of imbalanced data classification, as well as methods of measuring the performance of algorithms. In Section 4 the applied algorithms and the method of parallelizing are discussed. Section 5 describes the used benchmark datasets and the results of experiments, and finally Section 6 contains conclusions.

2. Overview – classification of imbalanced data sets

The most popular methods for classifying imbalanced datasets make use of sampling in the form of over-sampling or under-sampling. Both methods change the distribution of data by eliminating the majority class instances or increasing the minority class. In such case there is danger of deleting some potentially useful data, or in case of over-sampling, increase the possibility of over-fitting. For example SMOTE (Synthetic Minority Oversampling TEchnique) (Chawla, Bowyer, Hall, and Kegelmeyer, 2002) is a well-known method that creates new samples belonging to the minority class. The method is popular and different aspects were studied: choice of sampling rates, different methods of creating new data, making use of all samples in the process etc.

SMOTE together with genetic algorithms were used in Vannucci and Colla (2018). In this case SMOTE is used for both under-sampling and over-sampling, and the proportion rate is determined by the genetic algorithm. The original dataset is partitioned, keeping the imbalance ratio, into training (50%), validation (25%) and testing set (25%). The training set with the genetic algorithm is used to establish optimal rates, then the validation set is used for testing that stage, and finally best performing rates are applied in the final classification on the testing set.

A study on imbalanced enterprise credit evaluation appeared in Sun et al. (2018). It makes use of ensemble classifiers, SMOTE as a sampling technique and the Bagging algorithm with different sampling rates. The experiments were performed on financial data of 552 Chinese companies. The reported accuracies are around 0.85, but they are
not compared with other methods as there are no reported results for the considered dataset.

Under-sampling based on clustering is the method applied in Lin, Tsai, Hu, and Jhang (2017). The used procedure performs clustering to under-sample the majority class in order to balance it with the minority class. In this case no specialized under-sampling methods are used, only Bagging using produced clusters allows for balancing the training set. Two strategies for clustering are considered and the results for a large group of datasets are reported. Some are used as reference for our experiments (see Table 6).

Another group of methods is based on ensemble learning, for example, Bagging or Boosting. The SplitBal method considered in this paper is a variant of ensemble learning since for each so-called bin (see Section 4) a different classifier can be used.

An approach that gained some interest in the classification of imbalanced data makes use of the feature selection and feature extraction. The goal of the first one is to select a subset of features from the entire feature space to achieve optimal performance. On the other hand, feature extraction creates new features from the original ones using some functional mapping. Methods that make use of discriminative ability of features in the context of class imbalance data are considered in Zhang et al. (2015) and suggest among others one method EDBC which is further examined in the paper (Section 4.3).

SplitBal also belongs to a group of specialized algorithms where well-examined classification methods are adjusted to the case of imbalanced datasets. Another representative of that group of methods is the recently introduced KRNN – k-Rare-class Nearest Neighbour Classification (Zhang et al., 2017). The main idea of the reformulated k-nearest neighbour classifier is the introduction of the dynamic local query neighbourhood to ensure adequate presence of data from the minority class. The value of k is also changed dynamically. The report on results for datasets from Lichman (2013) show improvement of accuracies over the standard k-nearest neighbour method.

There is still another group of methods which in the process of training the classifier make use of the transformed confusion matrix so that different costs of misclassifying the samples from the majority and minority class are assumed.

For an up to date review of papers and ideas on the imbalanced data classification, see Haixiang et al. (2017) and Nguyen, Hwang, and Jung (2017).

3. Preliminaries

The classification task can be formulated as follows. Let $T$ be the dataset, $C$ – the set of classes:

$$T = \{(d, c) \mid d \in D, c \in C\} \subset D \times C,$$

$D$ is the space of attribute vectors $d = (w_1^d, \ldots, w_N^d)$ with $w_i^d$ being symbolic or numeric values. The learning algorithm looks for the best possible approximation $\hat{f}$ of the unknown function $f$ such that $f(d) = c$. Then $\hat{f}$ (classifier) can be used to find the class $c = \hat{f}(\bar{d})$ for any $\bar{d} \in T \mid D$. In case of imbalanced (binary) classification, the elements of
the dataset are unevenly distributed among the classes. If $c_1$ is the majority class, then the number of attribute vectors with class label $c_1$ highly outnumbers those from the minority class with label $c_2$:

\[ ||T_{c_1} = \{ d | \langle d, c_1 \rangle \in T \} || \gg ||T_{c_2} = \{ d | \langle d, c_2 \rangle \in T \} || \]

To measure the quality of classification, i.e., how well $\hat{f}$ approximates the unknown function $f$, the usual approach is to count the percentage of correctly classified observations taken from the set different than the one used for training, but with known class labels. However, in case of skewed distributions this may not work correctly since the minority class may hold a minimum effect on the overall accuracy. Therefore, other measures are recommended. Let the testing set be partitioned into four subsets: TP – true positive, TN – true negative, FP – false positive and FN – false negative. Define true-positive rate and false-positive rate as follows:

\[ TP_{rate} = \frac{TP}{TP + FN}, \quad FP_{rate} = \frac{FP}{FP + TN} \]

then AUC is the area under the ROC (Receiver Operating Characteristic) curve demonstrates the trade-off between the benefits ($TP_{rate}$) and costs ($FP_{rate}$). The points on the ROC curve are defined by ($FP_{rate}$, $TP_{rate}$) taken for different values of the classifier's threshold.

4. Proposed approach

This paper extends the results of Jedrzejowicz et al. (2017) where two methods, known from literature and potentially suited for parallelization, were the main issue. They were the splitting-based data balancing method (SplitBal) introduced in Sun et al. (2015) and the dissimilarity-based imbalance data classification (EDBC) introduced in Zhang et al. (2015).

For this paper SplitBal is the core idea, as different algorithms suggested in the paper use SplitBal at some stages. Therefore, it is discussed first.

4.1. Parallel version of SplitBal

The main idea of SplitBal (Sun et al., 2015) is to divide the majority class instances into several bins so that each bin contains the minority class and a part of majority class of the equal size. Thus, the number of bins $K$ is approximately equal to the imbalance ratio $IR = \frac{\text{number of majority class instances}}{\text{number of minority class instances}}$. After the training dataset is divided into bins, for each bin an independent training takes place and base, binary probabilistic classifiers $CL_1, \ldots, CL_K$ are evolved. These binary classifiers are further combined into an ensemble to classify new data. Applying parallelization to the used algorithm emerged as a natural step as follows from the above description. Training of different bins can be performed in parallel since no data exchange among computations for different bins is needed. What is more parallelization allows to perform experiments on relatively big datasets. Also the computation time can be significantly reduced in case of parallel experiments.
In our approach for parallel SplitBal, the MapReduce paradigm was used. The Map step includes: dividing the training set into bins, for each bin training a classifier on the training set is restricted to the bin and then the evolved classifier is applied to each instance from the testing set. In the Reduce step the results of the classifiers are merged to evolve the decision of the ensemble. To present the ensemble rules, the following notation is assumed. Let \( r \) stand for an instance from the testing set which is classified by the \( i \)-th classifier with probability \( P_{ij} \) as \( c_j \), for \( j = 1, 2 \). Further let \( D_{ij} \) stand for the average distance of \( r \) from the data with class label \( c_j \) in the \( i \)-th bin. The following ensemble rules allow to calculate final classification results \( R_1 \) and \( R_2 \) as shown below.

- **MaxDistance**
  \[
  R_1 = \arg \max_{1 \leq i \leq K} \frac{P_{i1}}{D_{i1} + 1}, \quad R_2 = \arg \max_{1 \leq i \leq K} \frac{P_{i2}}{D_{i2} + 1}.
  \]

- **MinDistance**
  \[
  R_1 = \arg \min_{1 \leq i \leq K} \frac{P_{i1}}{D_{i1} + 1}, \quad R_2 = \arg \min_{1 \leq i \leq K} \frac{P_{i2}}{D_{i2} + 1}.
  \]

- **ProDistance**
  \[
  R_1 = \prod_{i=1}^{K} \frac{P_{i1}}{D_{i1} + 1}, \quad R_2 = \prod_{i=1}^{K} \frac{P_{i2}}{D_{i2} + 1}.
  \]

- **MajDistance**
  \[
  R_1 = \sum_{i=1}^{K} \sgn(P_{i1}, P_{i2}) \frac{P_{i1}}{D_{i1} + 1}, \quad R_2 = \sum_{i=1}^{K} \sgn(P_{i2}, P_{i1}) \frac{P_{i2}}{D_{i2} + 1},
  \]
  where \( \sgn \) is the sign function:
  \[
  \sgn(x, y) = \begin{cases} 
  1 & \text{if } x \geq y, \\
  0 & \text{otherwise}.
  \end{cases}
  \]

- **SumDistance**
  \[
  R_1 = \sum_{i=1}^{K} \frac{P_{i1}}{D_{i1} + 1}, \quad R_2 = \sum_{i=1}^{K} \frac{P_{i2}}{D_{i2} + 1}.
  \]

Finally, the instance \( r \) is classified as \( c_1 \) if \( R_1 \geq R_2 \), otherwise as \( c_2 \).

The details of the algorithm are given as Algorithm 1.

Referring to experiments they are parametrized by:

- dataset with parameters: number of instances, number of attributes, imbalance ratio;
- the choice of a base classifier;
- the choice of ensemble rule: MaxDistance, MinDistance, ProDistance, MajDistance, SumDistance.
When performing the experiments, it was observed that the Reduce step took a definitely longer time than Map. Therefore, further speed-up was introduced by running in parallel computations performed in the Reduce step. Besides, we also exploited in-memory data structure store used as a database. This allowed to definitely decrease the processing time. Since SpliBal is used in our study in several examined algorithms, in Figure 1 it is graphically presented. Procedures which are optional in some algorithms are denoted with ∗.

4.2. Feature selection

Feature selection strategies are broadly applied in a preprocessing stage of classification. As noted in Nguyen et al. (2017) the preprocessing of imbalanced data performed before

---

**Algorithm 1: Algorithm SplitBal-parallel version**

**Input:** imbalanced training data \( TR \), \( C = \{c_1, c_2\} \) set of classes, \( N \) is the number of attributes. \( TS \) - testing data, \( K \) - number of bins, \( X \)- ensemble rule.

**Output:** quality of classification in the form of AUC - area under ROC curve

1. divide the training set \( TR \) into \( K \) balanced bins \( B_1, \ldots, B_K \)

   /* MAP step */

    2. for \( k = 1 \) to \( K \) do

        3. build the classifier \( CL_k \) for training data \( B_k \)

        4. for \( i = 1 \) to \( |TS| \) do

            5. apply classifier \( CL_k \) to \( i-th \) instance \( <d_i, cf_i> \in TS \)

            6. let \( C_{jk} \) stand for the probability assigned by classifier \( CL_k \) to instance \( d_i \)

                for class \( c_j \), for \( j = 1, 2 \)

    7. apply team classifier

        /* REDUCE step */

    8. for \( i = 1 \) to \( |TS| \) do

        9. using \( C_{jk} \) for \( j = 1, 2, k = 1, \ldots, K \) and strategy \( X \) find class \( l_i \)

        10. comparing \( l_i \) and \( cf_i \) update, respectively, the values of parameters: true positive, true negative, false positive and false negative

11. return AUC - area under ROC curve

---

When performing the experiments, it was observed that the Reduce step took a definitely longer time than Map. Therefore, further speed-up was introduced by running in parallel computations performed in the Reduce step. Besides, we also exploited in-memory data structure store used as a database. This allowed to definitely decrease the processing time. Since SpliBal is used in our study in several examined algorithms, in Figure 1 it is graphically presented. Procedures which are optional in some algorithms are denoted with ∗.

**4.2. Feature selection**

Feature selection strategies are broadly applied in a preprocessing stage of classification. As noted in Nguyen et al. (2017) the preprocessing of imbalanced data performed before

---

**Figure 1. Model of algorithms using SplitBal.**
classification boosts the results, allows to reduce noisy data and decreases the training
time. The task of feature selection is to select a subset of features from the entire
feature space that allows the classifier to achieve optimal performance.

The algorithm Relief (Kira and Rendell, 1992) for feature selection is very well suited for
parallelization. The idea is to introduce weights for attributes so that in an iterative
process a random vector \(x\) is chosen from the training set and two vectors \(H(x)\)
(nearest hit–nearest to \(x\) from the same class as \(x\)) and \(M(x)\) (nearest miss–nearest to
\(x\) from a different class) are found. If for an attribute, the values of \(x\) and \(H(x)\) differ
(which means that the attribute separates two instances from the same class and this
is undesirable), then the weight of that attribute is decreased. On the other hand, if
the values of \(x\) and \(M(x)\) differ (which means that the attribute separates two instances
from different classes), then the weight of the attribute is increased. The details are in
Algorithm 2.

Algorithm 2: Algorithm Relief

```
Input: \(TR = \bigcup_{c \in C} TR_c\), \(C\) - set of classes,
\(TR_c = \{<d, c> | d \in D\}\), where \(D\) is the space of attribute vectors
\(d = (w_1^d, \ldots, w_N^d)\), \(N\) is the number of attributes, \(J\) number of iterations,
\(\tau > 0\) - threshold for selecting attributes

Output: selected \(n\) attributes \(a_{i_1}, \ldots, a_{i_n}, n < N\)

1 for \(m = 1\) to \(N\) do
   \(A_m = 0\)
   /* \(A_i\) are weights of attributes */
3 for \(j = 0\) to \(J\) do
4 select randomly \(<d, c> \in TR\)
5 let \(H \in TR\) vector nearest to \(d\) from class \(c\)
6 let \(M \in TR\) vector nearest to \(d\) from class different than \(c\)
7 for \(m = 1\) to \(N\) do
8 \(A_{m,+} = -\frac{\text{diff}(m,d,H)_j}{\text{diff}(m,d,M)_j}\)
9 where \(\text{diff}(m,x,y)\) is the difference between \(x\) and \(y\) for the \(m\)-th
   attribute
10 for \(m = 1\) to \(N\) do
11 if \(||A_m|| > \tau\) then
12 select attribute \(m\) as output
13 /* attributes above threshold are selected */
13 return selected attributes
```

In our study Relief is used in two tested algorithms. In the first case, Relief is applied in a
preprocessing stage before SplitBal to check the impact of feature reduction on the classifi-
cation accuracy as well as on the processing time. This version is further referenced as
RelSplitBal. In Table 1 the impact of different values of features reduction on the classifi-
cation accuracy is shown. When the jump up in accuracy is observed, the names of attribu-
tes are cited. It is further discussed Section 5.
4.3. EDBC

Relief is also used in the algorithm EDBC – Expanded Dissimilarity-based Classification defined in Zhang et al. (2015). In our previous work it was examined, but the results and the processing times were much inferior, here some modifications are introduced. EDBC makes use not only of feature selection but also of two strategies well founded in data mining, that is prototype selection and data transformation based on dissimilarity calculation.

Algorithm 3: Algorithm EDBC

| Input: | imbalanced training data $TR = \bigcup_{c \in C} TR_c$, $C$ - set of classes, $TR_c = \{< d, c > | d \in D \}$, where $D$ is the space of attribute vectors $d = (d_1^d, \ldots, d_N^d)$ with $d_i^d$ being a numeric value and $N$ is the number of attributes. $TS$ - testing data, $\tau_1 > 0$ - threshold for clustering-based feature selection, $k$ - number of nearest neighbors, $\tau_2 > 0$ - threshold for selecting prototypes, $dis$ - dissimilarity measure. |
| Output: | quality of classification in the form of AUC - area under ROC curve |

*/* feature selection step */ *

1 perform feature selection on $TR$ (using RELIEF, Algorithm 2) and filter onto $TR' = \bigcup_{c \in C} TR'_c$ using threshold $\tau_1$,

*/* prototype selection step */ *

2 for each class $c \in C$ do
3 perform Jarvis-Patrick, or c-means, clustering on $TR'_c$ with parameter $k$,
4 using threshold $\tau_2$ select prototypes of the clusters for the class $P_c = \{p_1^c, \ldots, p_k^c\}$,

*/* transformation step */ *

5 for each class $c \in C$ do
6 for each $x \in TR'_c$ do
7 perform dissimilarity transformation on $x$ into $\tilde{x} = (dis(x, p_1^c), \ldots, dis(x, p_k^c))$,
8 use $\{\tilde{x} : x \in TR'_c\}$ as a training set for the classifier,
9 perform testing on the data set $TS$,
10 return AUC - area under ROC curve

The core of prototype selection is to perform clustering and then extract representative instances from clusters as prototypes. Data transformation is used to project data into dissimilarity space (see Algorithm 3 line 7) to improve discriminant ability of features via dissimilarities among examples. Originally, the authors of EDBC suggested to use Jarvis
Patrick clustering (Jarvis and Patrick, 1973) which is highly time consuming. It requires calculating the similarity of any two instances which is the number of k-shared nearest neighbours. The time complexity of this step is $O(n^3)$ for $n$ the size of the dataset, which is difficult for large-scale datasets. We made experiments both with Jarvis Patrick clustering and c-means clustering.

The second method allowed to speed up the computation time approximately twice, unfortunately for some datasets the results of the classification worsened which are shown in Table 5. It is worth noting that in case of numerical data classification, by features we usually mean attributes. It is different, for example, in text classification – ‘feature’ is usually a broader concept than ‘attribute’. The algorithm EDBC is described in Algorithm 3. In Section 5, where the experiments are described, it is denoted as EDBC.

### 4.4. Algorithms making use of correlation measure and sampling

Two other experiments were planned and performed for the survey. The first one uses correlation between data from the majority class and minority class, see Algorithm 4. After calculating the correlation of rows from the majority class, the algorithm is run in two versions:

- the sorted rows from the majority class are divided into bins and SplitBal is performed – this version is denoted CorrSplit,
- only the bin with rows best correlated with the minority class is used for building the classifier – this algorithm is denoted Corr.

In order to check whether oversampling the imbalanced data will give results comparable with SplitBal, the experiment using SMOTE (Chawla et al., 2002) was performed. This algorithm works in two steps: first over-sampling is performed to balance the minority class with the majority class, and then the classifier is trained. This algorithm is denoted as Smote in what follows.

**Algorithm 4:** Algorithm correlation based

```plaintext
Input: training data $TR = T_{c_1} \cup T_{c_2}$, where $T_{c_1}$-majority set, $T_{c_2}$- minority set, $T_{c_1} \gg T_{c_2}$, testing set $TS$, classifier version $cl$

Output: quality of classification $q_1$ (CorrSplit), $q_2$ (Corr) in the form of AUC

1 for $x \in T_{c_1}$ do
2    for $y \in T_{c_2}$ do
3        calculate $corr(x, y)$ the value of correlation between x and y
4        $c_x = \sum_{y \in T_{c_2}} corr(x, y)$
5    let $T^{sort}$ stand for $T_{c_1}$ sorted according to decreasing values of $\{c_x\}$
6    build classifier $L_1$ using SplitBal with classifier $cl$ and training set $T_{c_1}^{sort} \cup T_{c_2}$
7    let $q_1$ area under the ROC curve for classifier $L_1$ and testing set $TS$
8    let $T^{best}$ stand for the first $||T_{c_2}||$ rows of $T_{c_1}^{sort}$
9    classifier $L_2$ of type $cl$ using training set $T_{c_2}^{best} \cup T_{c_2}$
10   let $q_2$ area under the ROC curve for the classifier $L_2$ and testing set $TS$
11 return $q_1$, $q_2$
```
5. Experiments and datasets

To assess the performance of the proposed approach, we tested it over a set of publicly available benchmark datasets obtained from KEEL (Alcalá-Fdez et al., 2009) and UCI (Lichman, 2013). Repository KEEL contains a special section containing 145 datasets for imbalanced classification. Used datasets range from highly imbalanced (imbalance ratio IR = 129.44) to low imbalanced (IR = 5.9). The details on the datasets are given in Table 2. The table in each row contains the name of the data set, its origin, number of attributes, number of instances and imbalance ratio.

In the course of SplitBal implementation, the MapReduce paradigm was applied. MapReduce is proposed by Google as a programming model to process large and scalable datasets by parallel and distributed algorithms. The two major steps that is Map and Reduce are natural for the used SplitBal algorithm – the first is used to process each bin with a map and the second to collect the results. Experiments were conducted in software environment including interpreter of R language and Java Platform as well as Mongo as a document database. Scripts in R were used for the preprocessing task, dividing data into training and testing subsets as well as further distinguishing bins from testing data.

After some experiments with Naive Bayes, Logistic regression and C5.0 decision tree classifier, the latter was chosen. This classifier is further referenced (Table 5) as ‘Weak’ – this is in accordance with usual presentation of ensemble classifiers. From R package (R Core Team, 2013), we used C50 as well as CORElearn and the ROCR package for scoring the classifiers.

For the sake of MapReduce paradigm, the program in Java language was implemented. It precisely controls the number of processes dedicated for computations in both Map and Reduce steps. These numbers belong to the parameters of the computations. During the computations Java threads run and control operating system processes which perform computations as such, implemented as R language scripts. The Map step parallelization is straightforward and consists of performing independent computations for each bin. The Reduce step is a bit more complicated and relies on computing sums for pairs as they appear in the queue of already finished computations. The Java SE BlockingQueue class was used as a mechanism to synchronize the processes and their results. If the number of processes available to run is set to 1, then the computations will run sequentially, one after another. It allows to simulate well the sequential computations for SplitBal. The experiments were conducted on a 32 core processor machine and during the experiments approximately 15 cores were used (the number 15 was used as the parameter of number of available threads/processes for both MapReduce steps).

Table 2. Benchmark datasets used in the experiment.

| Dataset   | Origin                  | Attributes | Instances | Imbalance ratio |
|-----------|-------------------------|------------|-----------|-----------------|
| abalone19 | Alcalá-Fdez et al. (2009) | 8          | 4174      | 129.44          |
| abalone9-18 | Alcalá-Fdez et al. (2009) | 8          | 731       | 16.40           |
| bank      | Lichman (2013)          | 17         | 45211     | 75.47           |
| churn     | Lichman (2013)          | 21         | 3333      | 5.9             |
| German    | Lichman (2013)          | 20         | 1000      | 2.33            |
| hepatitis | Lichman (2013)          | 19         | 155       | 3.84            |
| magic     | Lichman (2013)          | 11         | 19020     | 61.35           |
| page block | Alcalá-Fdez et al. (2009) | 10         | 5472      | 8.79            |
| segment0  | Alcalá-Fdez et al. (2009) | 19         | 2308      | 6.02            |
| yeast6    | Alcalá-Fdez et al. (2009) | 8          | 1484      | 41.4            |
To measure the quality of classification on the benchmark datasets, AUC – the area under the ROC (Receiver Operating Characteristic) curve was used. This measure is recommended for the case of imbalanced datasets as the ROC curve is independent of class distribution and rather than calculating accuracy it makes use of the partition of the testing set into four subsets: TP – true positive, TN – true negative, FP – false positive and FN – false negative. The results of experiments for SplitBal are shown in Table 3. The table in each row contains the name of the data set, the name of the applied ensemble rule and the mean AUC value obtained in 30 experiments.

In Table 4 we compare the results of SplitBal with the results of SplitBal preceded by Relief to reduce the number of attributes and choose most valuable. The experiments for RelSplitBal were conducted for different values of the threshold, in Table 4 we report one experiment for some datasets. For the two algorithms and each dataset we give the number of attributes used, the quality of classification as AUC and the mean processing time. It can be observed that even decreasing the number of attributes by 50% does not change the quality of classification. We found interesting results for two big datasets: abalone19 and magic (see Table 1). In both cases, applying Relief and reducing the number of attributes gave worse results than for the whole set of attributes, but there is a jump up (3 attributes for abalone 19, 4 attributes for magic) which shows that some attributes are more distinctive than other. For these cases, the names of attributes are cited.

The comparison of results of all examined algorithms is given in Table 5. The results are grouped in two sets. The first contains those using SplitBal (i.e SplitBal and CorrSplit) and the second group algorithms not using the partition into bins. It is worth noting that SplitBal boosts the results considerably – the results in column 5 correspond to applying the classifier to unbalanced dataset (one bin) and the results in column 3 for SplitBal (number of bins equal to imbalance ratio) are much better in each case.

### Table 3. Classification results for SplitBal.

| Dataset     | Ensemble rule | AUC area |
|-------------|---------------|----------|
| abalone19   | SumDistance   | 0.825    |
| abalone9-18 | SumDistance   | 0.820    |
| bank        | SumDistance   | 0.895    |
| churn       | MaxDistance   | 0.92     |
| German      | SumDistance   | 0.750    |
| hepatitis   | MaxDistance   | 0.897    |
| magic       | SumDistance   | 0.871    |
| page block  | SumDistance   | 0.977    |
| segment0    | SumDistance   | 0.998    |
| yeast6      | MaxDistance   | 0.969    |

### Table 4. SplitBal versus RelSplitBal.

| Dataset     | No. att | AUC     | Time  | No. att | AUC     | Time  |
|-------------|---------|---------|-------|---------|---------|-------|
| abalone19   | 8       | 0.825   | 3 m   | 3       | 0.805   | 2 m58 s|
| bank        | 17      | 0.895   | 5 m32 s| 9       | 0.89    | 3 m18 s|
| churn       | 21      | 0.92    | 0 m10 s| 16      | 0.9     | 0 m7 s |
| magic       | 11      | 0.871   | 2 m40 s| 7       | 0.839   | 1 m44 s|
| page block  | 10      | 0.977   | 0 m19 s| 6       | 0.97    | 0 m11 s|
| segment0    | 19      | 0.998   | 0 m14 s| 10      | 0.99    | 0 m8 s |
| yeast6      | 8       | 0.969   | 1 m13 s| 3       | 0.92    | 0 m54 s|
The results of EDBC are poor. Not only the quality of classification is inferior compared to other methods but also the processing time was relatively longer. The comparison of processing times of the algorithm EDBC and SplitBal is given in Table 7. The processing time for EDBC and big data sets was very long, for example, for magic data it was longer than 12 hours and bank data longer than 24 hours, the results for Jarvis Patrick clustering are omitted and marked * in Tables 5 and 7. The results for c-means clustering are shown in brackets.

In Table 6 we compare our best results with those found in the literature with the source of results reported for reference. It follows that in several cases our results are superior, probably thanks to careful fine tuning of the parameters.

### Table 5. Classification results in terms of AUC.

| Dataset       | Imb. ratio | SplitBal | CorrSplit | 'Weak' | Smote | EDBC | Corr |
|---------------|------------|----------|-----------|--------|-------|------|------|
| abalone19     | 129.44     | 0.825    | 0.825     | 0.69   | 0.674 | 0.65 (0.5) | 0.699 |
| abalone9-18   | 16.40      | 0.820    | 0.816     | 0.725  | 0.733 | 0.68 (0.63) | 0.733 |
| bank          | 75.47      | 0.895    | 0.895     | 0.72   | 0.826 | * (0.5)    | 0.814 |
| churn         | 5.9        | 0.92     | 0.92      | 0.85   | 0.887 | 0.67 (0.65) | 0.856 |
| German        | 2.33       | 0.75     | 0.75      | 0.690  | 0.716 | 0.71 (0.70) | 0.716 |
| hepatitis     | 3.84       | 0.897    | 0.897     | 0.89   | 0.905 | 0.67 (0.64) | 0.905 |
| magic         | 61.35      | 0.871    | 0.645     | 0.76   | 0.753 | * (0.5)    | 0.645 |
| page block    | 8.79       | 0.977    | 0.894     | 0.91   | 0.943 | 0.76 (0.83) | 0.894 |
| segment0      | 6.02       | 0.998    | 0.979     | 0.99   | 0.983 | 0.95 (0.95) | 0.997 |
| yeast6        | 41.4       | 0.969    | 0.965     | 0.92   | 0.95  | 0.76 (0.72) | 0.965 |

### Table 6. Comparison of results.

| Dataset       | Our best result | Literature reported result | Source |
|---------------|-----------------|-----------------------------|--------|
| abalone19     | 0.825           | 0.728                       | Lin et al. (2017) |
| abalone9-18   | 0.820           | 0.831                       | Lin et al. (2017) |
| bank          | 0.895           | 0.89                        | Sharma, Kaur, Gandotra, and Sharma (2015) |
| churn         | 0.92            | 0.9194                      | Zhu et al. (2017) |
| German        | 0.75            | 0.743                       | Zhang et al. (2017) |
| hepatitis     | 0.905           | 0.873                       | Zhang et al. (2017) |
| magic         | 0.871           | 0.86                        | Inaba, Salles, Perron, and Caporossi (2018) |
| page block    | 0.977           | 0.991                       | Sun et al. (2015) |
| segment0      | 0.998           | 0.996                       | Lin et al. (2017) |
| yeast6        | 0.969           | 0.9465                      | Sun et al. (2015) |

### Table 7. Processing time.

| Dataset       | EDBC          | SplitBal       |
|---------------|---------------|----------------|
| abalone19     | 46 m18 s      | 0 m59 s        |
| abalone9-18   | 0 m52 s       | 0 m2 s         |
| bank          | * (36 h)      | 1 m10 s        |
| churn         | 71 m44 s      | 0 m10 s        |
| German        | 6 m9 s        | 0 m2 s         |
| hepatitis     | 0 m9 s        | 0 m2 s         |
| magic         | * (45 m36 s)  | 0 m46 s        |
| page block    | 1 h15 m8 s (43 m42 s) | 0 m8 s |
| segment0      | 8 m35 s       | 0 m7 s         |
| yeast6        | 4 m32 s       | 0 m21 s        |
6. Conclusions and future work

Further experiments with SplitBal as compared with our previous work proved that this algorithm is worth experimenting since the results are encouraging. Our environment developed for the experiments is easily scalable and since the results with RelSplitBal proved not to be worse than those for SplitBal some new experiments are planned with really big datasets.

Also new ideas connected with statistical preprocessing will be further examined since the results for Corr are tempting. For all examined datasets they were better than for EDBC. As for the algorithm EDBC which is parametrized by clustering, experiments with other clustering methods will be considered – for example usually fuzzy c-means is superior to c-means.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Joanna Jedrzejowicz is graduate of Warsaw University in mathematics, she obtained her Ph.D at Gdansk University, Poland and habilitation at Poznan University, Poland in informatics. Presently she is a professor of informatics in Faculty of Mathematics, Physics and Informatics, Gdansk University, Poland. Her research interests include artificial intelligence, complexity theory and formal languages and in this area she has published over 40 papers in the international scientific journals and proceedings, two books and several textbooks for students. She is working as program committee member in a number of international scientific conferences.

Robert Kostrzewski received the B. Sc in informatics in Faculty of Mathematics, Physics and Informatics, Gdansk University. He is currently working on his masters’s thesis in informatics.

Jakub Neumann holds Ph.D in mathematics from the University of Gdansk. He is an associate professor of informatics in Faculty of Mathematics, Physics and Informatics, Gdansk University. His current research interests include software engineering and machine learning. He has been leading several research projects in cooperation with IT firms dealing with e-learning.

Magdalena Zakrzewska is a doctoral student in Institute of Informatics, Faculty of Mathematics, Physics and Informatics, Gdansk University. Her research interests include machine learning and statistical natural language processing. She has Master’s degree in informatics from both University of Gdansk and Gdansk University of Technology.

ORCID

Joanna Jedrzejowicz http://orcid.org/0000-0003-4979-5476

References

Alcalá-Fdez, J., Sánchez, L., García, S., del Jesús, M. J., Ventura, S., i Guiu, J. M. G., & Herrera, F. (2009). KEEL: A software tool to assess evolutionary algorithms for data mining problems. Soft Computing, 13(3), 307–318. doi:10.1007/s00500-008-0323-y

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority oversampling technique. The Journal of Artificial Intelligence Research, 16, 321–357. doi:10.1613/jair.953.
Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. Expert Systems with Applications, 73, 220–239. Retrieved from http://www.sciencedirect.com/science/article/pii/S0957417416307175

Inaba, F. K., E. O. T. Salles, Perron, S., & Caporossi, G. (2018). Dgr-elm-distributed generalized regularized elm for classification. Neurocomputing, 275, 1522–1530. Retrieved from http://www.sciencedirect.com/science/article/pii/S0925231217316193

Jarvis, R. A., & Patrick, E. A. (1973). Clustering using a similarity measure based on shared near neighbors. IEEE Transactions on Computers. Institute of Electrical and Electronics Engineers, 22(11), 1025–1034. doi:10.1109/T-C.1973.223640

Jedrzejowicz, J., Neumann, J., Synowczyk, P., & Zakrzewska, M. (2017, July). Applying map-reduce to imbalanced data classification. In P. Jedrzejowicz, T. Yildirim, I. Czarnowski (Eds.), IEEE international conference on innovations in intelligent systems and applications, INISTA 2017, Gdynia, Poland, July 3–5 (pp. 29–33). IEEE. doi:10.1109/INISTA.2017.8001127

Kira, K., & Rendell, L. A. (1992, July). The feature selection problem: Traditional methods and a new algorithm. In W. R. Swartout (Ed.), Proceedings of the 10th national conference on artificial intelligence, San Jose, CA, July 12–16 (pp. 129–134). AAAI Press / The MIT Press. Retrieved from http://www.aaai.org/Library/AAAI/1992/aaai92-020.php

Lichman, M. (2013). Uci machine learning repository. Retrieved from http://archive.ics.uci.edu/ml/.

Lin, W.-C., Tsai, C.-F., Hu, Y.-H., & Jhang, J.-S. (2017). Clustering-based undersampling in class-imbalanced data. Information Sciences, 409–410(Supplement C), 17–26. Retrieved from http://www.sciencedirect.com/science/article/pii/S0020025517307235

Nguyen, T. T., Hwang, D., & Jung, J. J. (2017). Handling imbalanced classification problem: A case study on social media datasets. Journal of Intelligent and Fuzzy Systems, 32(2), 1437–1448. doi:10.3233/JIFS-169140

R Core Team (2013). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from http://www.R-project.org/

Sharma, N., Kaur, A., Gandotra, S., & Sharma, B. (2015). Evaluation and comparison of data mining techniques over bank direct marketing. International Journal of Innovative Research in Science, Engineering and Technology, 4(8), 7141–7147.

Sun, J., Lang, J., Fujita, H., & Li, H. (2018). Imbalanced enterprise credit evaluation with dte-sbd: Decision tree ensemble based on smote and bagging with differentiated sampling rates. Information Sciences, 425, 76–91. Retrieved from http://www.sciencedirect.com/science/article/pii/S0020025517310083

Zhang, X., Song, Q., Zhu, X., Sun, H., Xu, B., & Zhou, Y. (2015). A novel ensemble method for classifying imbalanced data. Pattern Recognition, 48(5), 1623–1637. doi:10.1016/j.patcog.2014.11.014

Zhang, X., Song, Q., Wang, G., Zhang, K., He, L., & Jia, X. (2015). A dissimilarity-based imbalance data classification algorithm. Applied Intelligence, 42(3), 544–565. doi:10.1007/s10489-014-0610-5

Zhu, B., Baesens, B., & vanden Broucke, S. K. L. M. (2017). An empirical comparison of techniques for the class imbalance problem in churn prediction. Information Sciences, 408(Supplement C), 84–99. Retrieved from http://www.sciencedirect.com/science/article/pii/S0020025517306618