Information Sciences 708 (2025) 122015

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

RIONIDA: A novel algorithm for imbalanced data combining instance-based learning and rule induction

Grzegorz Góra [a,*], Andrzej Skowron [b]

[a] University of Warsaw, Stefana Banacha 2, 02-097, Warszawa, Poland
[b] Systems Research Institute PAS, Newelska 6, 01-447, Warszawa, Poland

ARTICLE INFO

Keywords:
Imbalanced learning
Classification
Supervised learning
Instance-based learning
k nearest neighbours
Rule induction

ABSTRACT

The article presents the Rule Induction with Optimal Neighbourhood for Imbalanced Data Algorithm (RIONIDA) learning algorithm based on combination of two widely-used empirical approaches: rule induction and instance-based learning for imbalanced data classification. The algorithm is a substantial extension of the well-known the Rule Induction with Optimal Neighbourhood Algorithm (RIONA) learning algorithm developed for balanced data.
RIONIDA uses rules more general than in RIONA and realises a few additional concepts in comparison to RIONA, i.e. optimisation of the explicitly given performance measure defined over the confusion matrix, optimisation of weights for two classes, the idea of scaled rules, optimisation of parameters related to two latter ideas. RIONIDA, with decisions explainable for the user, is relatively fast and significantly outperforms the state-of-the-art algorithms analysed in the article.

1. Introduction

Imbalanced data analysis is a critical area of research in machine learning and statistics, particularly relevant in domains like medical diagnosis, fraud detection, and anomaly detection. In these contexts, the classes of interest (e.g. positive cases of a disease) are often significantly underrepresented compared to the majority class (e.g. healthy individuals). The current state of research encompasses the following techniques for handling imbalance: resampling methods, cost-sensitive learning, ensemble methods, evaluation metrics, deep learning, and generative models. The main current challenges are generalisation, overfitting, class distribution drift, lack of benchmarking, interpretability (explainability, XAI), and scalability. The reader is referred to the current literature dedicated to these topics (see e.g. [1,2]).

In this article, the problem of classifying imbalanced data is studied. Technically speaking, any data set used as an input for the classification problem in which examples of one class significantly outnumber the examples of the other class can be considered imbalanced (see e.g. [3,4]). However, the difficulty of learning from imbalanced data is not only related to the imbalance ratio between classes but mainly to the *data complexity* (see e.g. [4–6]). Generally speaking, the difficulty of imbalanced data is embodied in the complex structure of the minority class concept. The literature distinguishes several factors which make learning from imbalanced data a challenging task (see e.g. [4,6–8]). Among them are:

- selection of relevant performance measure (rather than accuracy),

* Corresponding author.
E-mail addresses: ggora@mimuw.edu.pl (G. Góra), skowron@mimuw.edu.pl (A. Skowron).

https://doi.org/10.1016/j.ins.2025.122015
Received 29 June 2024; Received in revised form 22 February 2025; Accepted 22 February 2025

Available online 25 February 2025

- relevant representation (in particular, searching for relevant features and using relevant similarity measure),
- data decomposition leading to *small disjuncts* (within-class imbalance between subconcepts),
- overlapping between the classes,
- presence of outliers or noisy examples,
- imbalance ratio,
- absolute number of examples.

Thus, the factor of the imbalance ratio is usually combined with the other factors mentioned above. The imbalance ratio can enhance the difficulty of those factors.

The standard classifiers (designed for balanced data) do not work well with imbalanced data for at least four reasons:

- Standard classifiers aim to maximise the classification *accuracy*. However, for imbalanced data, this performance measure is inadequate.
- The construction of standard classifiers in case of imbalanced data leads to achieving a rather low accuracy rate for the minority class while achieving high accuracy rate for the majority class (see e.g. [9]).
- Standard algorithms identifying noisy examples, i.e. training objects with incorrect decision labels, do not distinguish between the decisions labelling them into majority or minority classes. Suppose an example genuinely belonging to the minority class is identified as noisy, or a truly noisy example from the majority class is not identified as such. In that case, classifying objects from the minority class gets complicated (see e.g. [4]).
- Standard classifiers assume equal misclassification costs for all classes. However, the misclassification cost can be often much higher for the minority class than for the majority class.

Research on imbalanced data analysis is evolving rapidly, with various of methodologies being explored. However, significant challenges remain, particularly with respect to model generalisation, overfitting and interpretability. Overcoming these challenges is crucial for the successful application of machine learning in imbalanced scenarios in various domains.

In this paper, we present a novel algorithm called RIONIDA combining instance-based learning and rule induction methods to deal with the imbalanced data analysis. The aim was to develop an algorithm for dealing with imbalanced data characterised by (i) the high quality relative to the measure relevant for imbalanced data analysis and (ii) very good interpretability. The RIONIDA algorithm realises this aim. It is an extension of the RIONA algorithm [10,11].

RIONA was designed for balanced data analysis, optimises the accuracy performance measure and has good interpretability. RIONIDA, with decisions explainable for the user, is relatively fast and significantly outperforms the state-of-the-art algorithms analysed in the article, in particular on difficult regions regarding the analysis of minority class [5].

The presented approach departs from relying on pre-defined granules around test cases and their (partial) inclusion in decision classes. It utilises a more nuanced reasoning method for classifying objects in imbalanced datasets. This new approach involves a detailed analysis of training cases within the identified granules. Decision-making for a test case follows some specific processes. They can be roughly described as follows.

- *Neighbourhood Identification*: The k-Nearest Neighbor (kNN) method with a specialised distance measure and optimised k value identifies a relevant training case 'granule' (neighbourhood) of the test case.
- *Rule Creation*: A specific rule is created for a given test case and each training case within the neighbourhood.
- *Sub-granule Formation*: The 'left-hand sides' (conditional parts) of these rules are refined (scaled). Then, it is checked for each training case if this refined rule forms a sub-granule with training cases having the same decision class as the original training case.
- *Label Assignment*: If such a sub-granule is successfully formed, it strengthens the argument for assigning the same decision class to the test case (as it shares properties with the training case used to create the rule).
- *Optimal Weights*: The presented approach also supports the discovery of optimal weights for minority and majority classes, which have an impact on the label assignment.

RIONIDA takes into account a large space of parametrised rules, and the efficient optimisation of these meta-parameters is performed in the learning phase of RIONIDA. This paper presents the algorithm's code, which provides a new and effective method for solving the imbalanced data classification problem. RIONIDA is available in WEKA [12].

The novelty of this work is the development of the RIONIDA algorithm for imbalanced data:

- with high classification quality relative to the relevant measures,
- with very good interpretability by using human-understandable rules and instance-based learning,
- time efficient (among other things, due to the use of dynamic programming techniques and special data structures),
- significantly outperforming the algorithms compared to it,
- with provided computational complexity analysis,
- which is publicly available [12,13].

The article is a substantial extension of [14]. In particular, it includes a more detailed description of the RIONIDA algorithm, an explanation of the significance of RIONIDA's components, a more thorough description and presentation of experimental results, the experimental arguments that explain the very good performance of RIONIDA, and an analysis of RIONIDA's time performance.

The article is structured as follows. Section 2 discusses related works. Section 3 presents some basic concepts and notation used in the article. Section 4 presents the development of the RIONIDA algorithm, which consists of two main stages. In Subsection 4.1, motivation for the work is discussed. Subsection 4.2 describes the RIONA algorithm designed for balanced data. Subsection 4.3 introduces RIONIDA, a modification of RIONA, designed to classify imbalanced data. Subsection 4.4 shows how RIONIDA internal parameters can be learned very efficiently. Subsection 4.5 presents an estimation of the time and space complexities of RIONIDA. Section 5 describes the results of experiments showing that RIONIDA outperforms some state-of-the-art algorithms for imbalanced data on benchmarks and real-life data sets. Section 6 discusses some additional experiments helping to understand the advantages of RIONIDA and the reasons for its good performance. It also analyses the running time of RIONIDA and compares it to the other algorithms. Finally, Section 7 concludes the article.

2. Related work

In the past, there have been some attempts to combine instance- and rule-based approaches, but only for balanced data (see e.g. [15–18]).

The approach used in developing RIONIDA differs from the ones presented in the literature. To our knowledge, the only algorithm designed for imbalanced data analysis that combines the instance- and rule-based approaches and, at the same time, belongs to the algorithmic level approach (which modifies algorithms for balanced data) is BRACID (see e.g. [5]). BRACID is a modification of the RISE algorithm to make it applicable for imbalanced data. There are some substantial differences between BRACID and RIONIDA: (i) BRACID calculates rules in the learning phase (in advance), while RIONIDA does it in the testing phase (i.e. according to the lazy approach); (ii) BRACID starts from rules equivalent to instances and induces quasi-optimal rules for the given data set; RIONIDA adopts a different strategy and takes into account a large space of parametrised rules formulated in a specific language; different parametrisations correspond to different approaches, including (a) pure instance-based approach, (b) pure rule-based approach, and (c) approaches that combine them; for the given data set, RIONIDA selects the optimal parameter settings of rules, and does it very efficiently; (iii) BRACID optimises rules for F-measure, while RIONIDA can optimise any performance measure specified by a user (defined based on *confusion matrix*), and does it effectively (in comparison to direct searching of RIONIDA enormous parameter space).

RIONIDA is an extension of RIONA [10,11]. RIONA uses a lazy learning approach (see e.g. [19][17][18]), that is, it induces a very limited set of decision rules relevant only to the test example. The classification performed by RIONA on a given test object is based on rules induced only from a neighbourhood of the given test example. The empirical study showed that it is enough to consider a small neighbourhood to achieve classification accuracy comparable to the algorithm induced from the whole learning set. Moreover, it uses rules with conditions of the form: *attribute belongs to a set of values*. These sets of values are specified by grouping both numerical and symbolic values of attributes. In particular, RIONA does not require discretisation (or value grouping). In voting for the decision by rules *covering* the example being classified, the aggregation of the support sets of such rules is used. RIONA constructs object neighbourhoods of the optimal size. The learning of the optimal neighbourhood is based on the idea of dynamic programming (see e.g. [20] [pp.189-192]), which makes the computational time complexity of this step low.

RIONA was reported in the literature as one of the most accurate classification methods in many experimental comparisons done by various researchers to name a few (the authors use the most common RIONA implementation from WEKA platform named RseslibKnn): Facebook content recognition (RIONA was the best one of 21 tested algorithms), environmental sound recognition (best of 9 algorithms), metabolic pathway prediction of plant enzymes (2nd of 47 algorithms), acoustic-based environment monitoring (2nd of 8 algorithms), context awareness of a service robot (2nd of 8 algorithms), student performance prediction (5th of 47 algorithms). For literature describing the mentioned experimental comparisons, see [11].

One can also consider the Nested Generalised Exemplar (NGE) algorithm and its modifications [17,18] as examples of algorithms based on the combination of instance-based and rule-based approaches (they are classifying new data points by computing their distance to the nearest "generalised exemplar", i.e. either a point or an axis-parallel rectangle). However, the reported experimental results [18] show that NGE and its modifications are inferior to kNN on most of the tested data sets, while experiments with RIONIDA (extension of RIONA) show that RIONIDA outperforms kNN (with appropriate filters) on most of the tested data sets.

Deep learning techniques such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have recently been increasingly applied to imbalanced classification problems (see e.g. [1]), including applications in specific domains (see e.g. [2]). These techniques can learn complex patterns in the data and can be effective in dealing with class imbalance. Other techniques include generative adversarial networks (GANs) for data augmentation, meta-learning approaches, the use of reinforcement learning for imbalanced problem solving, exploring the strong links between imbalanced classification and emerging areas, such as federated learning, transfer learning, multitask learning, and an increased focus on the ethical and societal implications of imbalanced classification, particularly in high-stakes decision scenarios.

In RIONIDA, new simple features are generated by grouping symbolic and numerical attributes. Deep learning based imbalanced classification methods generate more compound new features of objects. In the future, we plan to extend RIONIDA, e.g. by allowing additional new features discovered by deep learning methods to be used to define distances for kNN. In this paper, we have discussed the advantages of RIONIDA concerning the problems of the joint effect of class imbalance and overlap, as well as explainability. These issues are also central to deep learning based imbalanced classification methods. In the case of RIONIDA, explainability is directly achieved through the use of human-understandable rules. In contrast, in the case of deep learning methods, explainability

requires serious additional work to be done. RIONIDA shows good performance on complex data. However, the joint effect of class overlap and imbalance in both cases (i.e. RIONIDA as well as deep learning based imbalanced classification methods) is still not fully understood and argues for the need to move towards a unified view of the class overlap problem in imbalanced domains. A more detailed analysis of this will be discussed in our following paper. Finally, we would like to mention another important feature of RIONIDA related to the explainability of the reasoning that leads to the construction of RIONIDA. For example, the argument that the decision for the test object *tst* is the same as for a given training object *trn* is based on the support of the (properly scaled) rule spanning *tst* and *trn*, and the aggregation of arguments across training objects is also easily understood by humans. In the case of methods based on deep learning, explaining the reasoning that leads to the construction of the target network is far from simple and requires the development of advanced new methods.

3. Basic notions

The domain of learning is a space of objects $\mathbf{X}$. Each object $x \in \mathbf{X}$ is described by a finite set of pairs $(a, a(x))$, where a is a conditional attribute from a given set A of (conditional) attributes, i.e. $a : \mathbf{X} \to V_a$ for $a \in A$, where the codomain V_a of a is the set of values of a and $a(x)$ is the value of a on the object $x \in \mathbf{X}$ [21]. We consider two types of attributes: numerical and symbolic. We denote the sets of symbolic and numerical attributes by A_{sym} and A_{num}, respectively.

Definition 1. Any tuple $(\mathbf{X}, A, d, \{\varrho_a\}_{a\in A})$, where $\mathbf{X}$ is the space of objects, A is a set of attributes, $d : \mathbf{X} \to V_d$ is a decision attribute ($d \notin A$), and for any attribute $a \in A$ is given metric[1] ϱ_a on the respective value set V_a (i.e. for any $a \in A$, (V_a, ϱ_a) is a metric space) is called *metric decision system*.

We use metrics for two reasons. First, they are used to construct metrics over objects. By default, we assume that the aggregated metric ρ is defined as the sum of individual metrics, denoted as $Agr(\{\varrho_a\}_{a\in A})$. Second, metrics are used for grouping attribute values in the construction of generalised rules, i.e. ρ_a is used to define for a given $v \in V_a$ a neighbourhood of similar values for v. RIONIDA as RIONA learns (as default setting) Simple Value Difference Metric (SVDM) metrics [15] for symbolic attributes; and for numerical attributes, uses Euclidean metric on $\mathbb{R}$.

Learning algorithms considered in the paper aim to induce classifiers of high quality (defining the high-quality approximations of decision functions) from restrictions of a given metric decision system to the set of training objects $trnSet \subset \mathbf{X}$.

In the case of a binary metric decision system for imbalanced data, we use the notation $V_d = \{d_{maj}, d_{min}\}$, where d_{min} represents the *minority class* and d_{maj} represents the *majority class*.

Definition 2. A *decision rule* over a metric decision system $(\mathbf{X}, A, d, \{\varrho_a\}_{a\in A})$ is an expression of the form $if\ t_1 \wedge t_2 \wedge \ldots \wedge t_m\ then\ d = v$, where $v \in V_d$, m is the number of attributes on the left-hand side of the rule, t_i is a condition of the form $a_i \in V$ (where $V \subseteq V_{a_i}$ is defined using ϱ_{a_i} and some parameters) for an attribute a_i $(i = 1, 2..., m)$.

For the training set $trnSet \subset \mathbf{X}$ and test example $tst \in \mathbf{X} \setminus trnSet$ we define $N(tst, trnSet, k, \varrho)$ (when used in the article with smaller arguments, others are fixed) as the set of k training examples from $trnSet$ that are most similar to tst according to the aggregated metric ϱ. We assume in this article that such neighbourhood N contains not more than k examples.[2]

4. Two stages for RIONIDA development

The RIONIDA algorithm is based on the combination of instance- and rule-based methods. Its development consists of two stages:

- the RIONA algorithm for balanced data and
- the RIONIDA algorithm based on a modification of the RIONA algorithm to make it relevant for imbalanced data.

Thus, RIONIDA belongs to the algorithmic-level approach among methods for learning from imbalanced data.

4.1. Motivations

In the past, there have been some attempts to combine instance- and rule-based approaches, however, only for balanced data (see e.g. [15,16]). Nonetheless, at least two reasons are advocating for developing such approaches not only for balanced but also for imbalanced data.

First, both approaches use reasoning schemes easily understandable by a human. Such schemes include rules in the form of

***If** some conditions are satisfied, **then** the decision is X*

[1] In fact, we use metrics or pseudometrics. For simplicity, we do not distinguish between them in this paper (in [11] this distinction is provided).

[2] In our implementation, when more than one example has the same distance from the object tst to the k-th nearest example, all of them are added to $N(tst, trnSet, k, \varrho)$. Theorems and analyses taking this into account are discussed in detail in [11]. We help the reader to easier catch the essence of considerations by simplifying them using the assumption made in this article.

which are often used by humans. Analogously, the reasoning scheme of the form

Since *our new example A*

is the most similar to *another known, examined example B,*

then *example A should have the same decision as example B*

used in instance-based learning is also easily understandable by humans. Because of this, such approaches meet the requirements for Machine Learning systems to be explainable. Together with the decision for the given test object, the classifying system should provide an explanation for this decision understandable by the user. In the last years, one can observe rapidly increasing importance of this issue in real-life applications. This is related to the topic of so-called Explainable Artificial Intelligence (see e.g. [22]).

Second, there are some intuitions, following from mathematical considerations, suggesting the use of instance-based learning, perhaps in combination with rule induction. The rule-based approaches are examples of a two-stage procedure. At the first stage, we induce (estimate) the unknown decision function. In the second stage, we apply this induced function to classify test examples. However, Vapnik observed that the decision function estimation is a much more general problem than we usually need to solve in practice. In most of the cases, we only need to estimate the unknown decision function at 'a few' new points defined by test objects (see [23, p. 12]). He suggests that if one needs to infer decisions for new cases based on small training sets, one should take into account the following principle:

> If you possess a restricted amount of information for solving some problem, try to solve the problem directly and never solve a more general problem as an intermediate step. It is possible that the available information is sufficient for a direct solution but is insufficient for solving a more general intermediate problem. [23]

This principle suggests that using instance-based approaches can be relevant. It also applies to methods combining instance-based approaches with other ones.

4.2. Preliminary work: the RIONA algorithm

In Section 2 we wrote about the RIONA algorithm which is our main reference work. However, here we present some more technical details about this algorithm since it can help the reader to better understand the RIONIDA algorithm presented in the paper. Also, the code of RIONIDA is based on the code of RIONA.

RIONA algorithm is presented in Algorithm 1.[3] It uses function $isConsistent\ (r, verifySet)$ *isConsistent(r, verifySet)*, which checks whether rule r is consistent with $verifySet$, i.e. if all the examples belonging to $verifySet$ satisfying the left-hand side of r are labelled by the same decision as the decision of r. This algorithm predicts the most common class among the training examples that are covered by the rules satisfied by a given test example and are in the specified neighbourhood. It uses g-rule, which informally is built of conditions chosen in such a way, that both the training and the test example satisfy the rule and the conditions are maximally specific. Its definition is generalised for RIONIDA. Thus, formally g-rule can be defined as sg-rule (see Definition 3) with fixed $s = 1$. For numerical attributes, the interval's endpoints are determined by the attribute values of the examples tst and trn used to form the rule. For symbolic attributes, the condition represents the specific group of values defined by a metric ball centered in the attribute value of the test example with the smallest radius such that the ball contains also training example.

Please note that in the mentioned definition, we only use metrics for symbolic attributes ($a \in A_{sym}$). However, in this definition, for the numerical attributes, the natural order between its values and the natural Euclidean metric are assumed (numerical values are grouped into intervals which represent grouping by using Euclidean metric).

For every decision class, the RIONA algorithm computes the support set restricted to the neighbourhood $N(tst, k)$. For every training object trn from the neighbourhood $N(tst, k)$ the algorithm constructs the rule

$$g\text{-}rule\left(tst, trn, \{\varrho_a\}_{a\in A_{sym}}\right)$$

based on the considered example trn and the test example tst. Then, it checks whether this g-rule is consistent with the remaining training examples from the neighbourhood $N(tst, k)$. If the local decision rule is consistent, then the training example trn used to construct the rule is added to the support set of the appropriate decision. Finally, the algorithm selects the decision with the support set of the highest cardinality.

It was proved in [11] that the $g\text{-}rule\left(tst, trn, \{\varrho_a\}_{a\in A_{sym}}\right)$ has a property that the examples distanced from the test example tst more than the training example trn cannot cause inconsistency of the rule. In consequence, to check the consistency of the rule with the $trnSet$, we could restrict consistency checking to the set $N(tst, trnSet, k, \varrho)$ (see line 9 of Algorithm 1). Moreover, using the mentioned property, if we sort N according to the distance $\varrho(trn, \cdot)$ obtaining $(nn_1, \ldots, nn_{|N|})$ then one could restrict checking the consistency of the rule built by tst and nn_i to the set $\{nn_1, nn_2, \ldots, nn_{i-1}\}$.

Below we describe the algorithm for estimation of the optimal value k for the neighbourhood $N(tst, k)$. This can be done in an analogous way to searching for the optimal value k for the kNN method. The leave-one-out method is used on a training set to

[3] The cardinality of the set X is denoted by $|X|$.

Algorithm 1: RIONA-classify(tst, $trnSet$, k, $\{\varrho_a\}_{a\in A}$).

Input: test example tst, training set $trnSet$, positive integer k, family of metrics for attributes $\{\varrho_a\}_{a\in A}$
Output: predicted decision for tst

begin
  $\varrho = Agr(\{\varrho_a\}_{a\in A})$
  $neighbourSet = N(tst, trnSet, k, \varrho)$
  foreach *decision* $v \in V_d$ **do**
    $supportSet(v) = \emptyset$
  end
  foreach $trn \in neighbourSet$ **do**
    $v = d(trn)$
    if $isConsistent\left(g\text{-}rule\left(tst, trn, \{\varrho_a\}_{a\in A_{sym}}\right), neighbourSet\right)$ **then**
      $supportSet(v) = supportSet(v) \cup \{trn\}$
    end
  end
  return $\arg\max_{v\in V_d} |supportSet(v)|$
end

estimate the Accuracy of the classifier for different values of k ($1 \le k \le k_{max}$); then the value of k with the highest estimated Accuracy is selected. Applying it directly would require repeating leave-one-out estimation k_{max} times. However, using dynamic programming technique, we emulate this process in time comparable to the single leave-one-out test for k equal to the maximal possible value $k = k_{max}$. Below we present the algorithm implementing this idea.

Algorithm 2: getClassificationVector(trn, $trnSet$, k_{max}, $\{\varrho_a\}_{a\in A}$).

Input: currently considered example $trn \in trnSet$, training set $trnSet$, number k_{max}, family of metrics for attributes $\{\varrho_a\}_{a\in A}$
Output: vector C of leave-one-out classification for trn for different values of parameter $k = 1, 2, \ldots, k_{max}$

begin
  $\varrho = Agr(\{\varrho_a\}_{a\in A})$
  $N = N(trn, trnSet \setminus \{trn\}, k_{max}, \varrho)$
  vector $(nn_1, \ldots, nn_{|N|}) = (N$ sorted by the distance $\varrho(trn, \cdot))$
  foreach *decision* $v \in V_d$ **do**
    $decStrength[v] = 0$
  end
  $currentDec =$ the most frequent decision in $trnSet$
  for $k = 1$ **to** $|N|$ **do**
    if $isConsistent(g\text{-}rule\left(trn, nn_k, \{\varrho_a\}_{a\in A_{sym}}\right), N)$ **then**
      $v = d(nn_k)$
      $decStrength[v] = decStrength[v] + 1$
      if $decStrength[v] > decStrength[currentDec]$ **then**
        $currentDec = v$
      end
    end
    $C[k] = currentDec$
  end
  return C
end

Algorithm 3: findOptimalK($trnSet$, k_{max}, $\{\varrho_a\}_{a\in A}$).

Input: training set $trnSet$, number k_{max}, family of metrics for attributes $\{\varrho_a\}_{a\in A}$
Output: optimal k

begin
  foreach $trn \in trnSet$ **do**
    $A_{trn} = getClassificationVector(trn, trnSet, k_{max}, \{\varrho_a\}_{a\in A})$
  end
  for $k = 1$ **to** k_{max} **do**
    $estimatedAccuracy[k] = \left|\{trn \in trnSet : d(trn) = A_{trn}[k]\}\right|$
  end
  return $\arg\max_{1\le k\le k_{max}} estimatedAccuracy[k]$
end

For a training example trn, the function $getClassificationVector(\ldots)$ (see Algorithm 2) finds k_{max} examples from $trnSet \setminus \{trn\}$ nearest to the example trn and sorts them according to the distance $\varrho(trn, \cdot)$ (i.e. we consider the metric ϱ with the first argument

fixed). Next, it returns the vector of decisions that the RIONA classifier would return for successive values of k. Algorithm 3 calls this routine for every training object, and then it selects the value k for which the global estimation of Accuracy is maximal.

Moreover, the Optimal Nearest Neighbour (ONN) algorithm is used, which instead of rules uses kNN method and similarly learns the optimal neighbourhood as RIONA does.

4.3. The RIONIDA algorithm

It turns out that RIONA has some drawbacks characteristic for the standard algorithms running on imbalanced data. Here comes the second step of the algorithm development. Now, the objective is to modify the proposed algorithm combining instance- and rule-based methods (RIONA) for improving its performance on imbalanced data. To simplify the task, the number of decision classes in RIONIDA is limited to only two (i.e. RIONIDA is directly applicable only for binary classification problems). All the ideas for RIONA are also realised in the RIONIDA algorithm. Moreover, this new algorithm realises a few new ideas in comparison with RIONA.

- RIONIDA does not try to maximise accuracy, but one of the performance measures that are much more relevant for imbalanced data, such as *F-measure* or *G-mean* (see e.g. [24,25]).
- Conflict resolution of rules in RIONIDA is more sophisticated than in RIONA. The aggregation of decisions of rules covering classified objects is defined using the property that the minority class is 'more important' than the majority class. The phrase 'more important' is expressed by the importance degree. The importance degree of the minority class (and in consequence of the majority one) is tuned during learning.
- Rules *inconsistent* to a certain degree are allowed. The level of inconsistency is also tuned during learning.

The following paragraphs introduce the main ideas behind the RIONIDA algorithm. In particular, below, we present the idea of more general rules in comparison to the ones used in RIONA, which enables to realise the third of the ideas of RIONIDA presented above.

Scaled generalised local decision rules (for short, the *sg-rules*) are used in RIONIDA, which are more general than rules used for RIONA. The idea is to select between rule-based method (as RIONA does) or kNN method (as the ONN algorithm does), and, on the other hand to allow a smooth transition between these approaches.

Definition 3. For any test object tst and any training object trn, we define the *sg-rule* denoted by $sg\text{-}rule\left(tst, trn, \{\varrho_a\}_{a\in A_{sym}}, s\right)$ or simply $sg\text{-}rule\,(tst, trn, s)$ (whenever parameters $\{\varrho_a\}_{a\in A_{sym}}$ are clear from the context or irrelevant due to generality), the decision rule with the decision $d(trn)$ and the following conditions t_a for each attribute a:

$$t_a = \begin{cases} a \in [a(tst), a(tst) + l \cdot s] & \text{when } s \geq 0, cond_1 \\ a \in [a(tst) - l \cdot s, a(tst)] & \text{when } s \geq 0, cond_2 \\ a \in B\left(a(tst), r_a \cdot s\right) & \text{when } s \geq 0, cond_3 \\ a \in \emptyset & \text{when } s < 0, \end{cases}$$

where $cond_1 \equiv a$ is numerical, $a(tst) \leq a(trn)$, $cond_2 \equiv a$ is numerical, $a(tst) > a(trn)$, $cond_3 \equiv a$ is symbolic, $l = abs(a(tst) - a(trn))$, $r_a = \varrho_a(a(tst), a(trn))$, and $B(c, R)$ is the closed metric ball of radius R centred at point c for metric ϱ_a, $s \in [-1, 1]$ is the scaling parameter of the rule, and abs denotes the absolute value.

Any sg-rule covers the test example, and the interval or ball corresponding to each attribute are scaled by the given parameter s in comparison to the rules used in RIONA.[4]

RIONIDA, compared to RIONA, additionally: (i) adds the possibility of choosing the performance measure to be optimised; in fact, measures more relevant for imbalanced data are taken into account (e.g. F-measure or G-mean), (ii) sets sensitivity constraint (for the minority class) to a higher level; furthermore, this sensitivity is adjusted to the currently analysed data, (iii) provides not only a possibility to learn when to use ONN (kNN like method) and when rule-based method, but also a combination of both types of algorithms (by tuning levels of rules inconsistency provided in Definition 3 a smooth transition between both types of algorithms is incorporated), (iv) automatically induces features not only those embedded in RIONA (optimal neighbourhood size and optimal metric), but also others. Fig. 1 shows the core idea of these algorithms and RIONIDA improvement over RIONA. Table 1 describes features of RIONA and RIONIDA showing their differences and RIONIDA improvement over RIONA.

In RIONIDA, after choosing the performance measure (which is relevant to the user's needs), the learning phase is performed relative to this chosen performance measure. In consequence, the same chosen performance measure is used both in training and testing. The internal parameters (size of the neighbourhood – parameter k, sensitivity to the minority class – parameter p, the degree to which the rules are used – parameter s) are learned during the learning phase. It is important to note that we present efficient in time methods of learning all of these parameters by the dynamic programming technique.

[4] For $s = 1$, this definition is equivalent to conditions used in RIONA. For $s = 0$ we have the rule covering only the test example and the training examples identical with the test example for all numerical attributes and distanced by 0 for all symbolic attributes. For $s < 0$, the premise of this rule is always false (formally speaking, not satisfied by any example) which relates to the elimination of consistency checking (see description of the function $isConsistent$ in Section 4.2) and in consequence to working as the kNN algorithm. The parameter s such that $0 < s < 1$ defines the scaling of the satisfiability area of the rule.

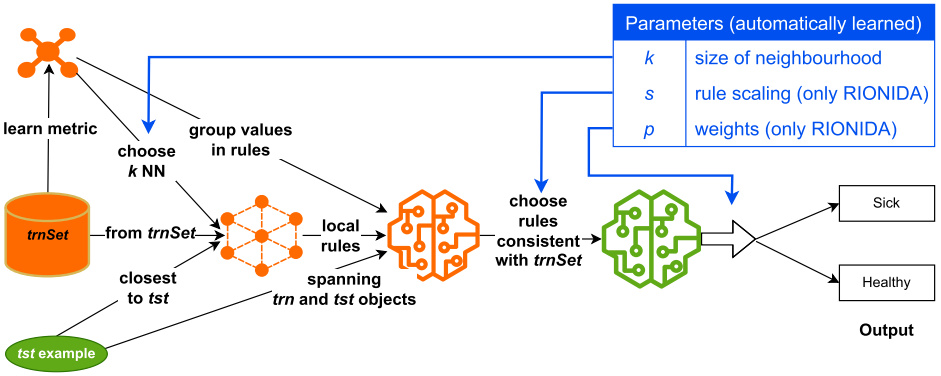

Fig. 1. Schematic diagram showing the core idea of RIONIDA (and RIONA) and the improvement over RIONA.

Table 1
Features of RIONIDA and RIONA algorithms and their differences. The signs ✓ and ✗ mean yes and no, respectively.

Features	RIONA	RIONIDA
Proper for imbalanced data	✗	✓
Proper for balanced data	✓	✓
Decisions explainable to the user	✓	✓
The reasoning behind the algorithm construction is explainable to the user	✓	✓
Combines rule- and instance-based learning	✓	✓
Lazy-based learning	✓	✓
Limits the neighbourhood to k closest objects (based on the learned metric)	✓	✓
Optimisation of the size of the neighbourhood (k)	✓	✓
Applying and optimisation of class weighting (p)	✗	✓
Applying scalable rules and optimisation of scalability parameter (s)	✗	✓
Users can specify the optimised quality measure, e.g. F-measure, G-mean or Accuracy	✗	✓
Space dimension of simultaneously optimised parameters	1	3
Multiclass classifier	✓	✗
Learning of the metric from training set	✓	✓
Automatic and quick grouping of numerical values (based on the learned metric)	✓	✓
Earlier discretisation not required	✓	✓
Automatic and quick grouping of symbolic values (based on the learned metric)	✓	✓
Rule consistency checking	✓	✓
Effective learning phase by using the idea of dynamic programming	✓	✓

For each parameter, there is a set of values that we take into account. The sets of admissible values for the parameters k, p, s, we denote by K, P and S, respectively. Thus, the set of possible classes of classifiers that we search for are of the cardinality $|K| \cdot |P| \cdot |S|$.

Parameter p is responsible for making the minority class more important than the majority one. This importance is expressed by the degree of importance of the minority class expressed by p. One can relate this to cost-sensitive learning. On the other hand, the parameter s is responsible for what kind of data we have: whether the data is more suitable for the rule-based classification, or maybe more for the kNN-type classification. This parameter allows us to fine-tune the type: completely kNN data, completely rule-based data, slightly rule-based data (e.g. 30%) and more kNN (e.g. 70%). It is the latter that corresponds to the smooth transition from the rule-based type of classifier to the kNN type classifier. Of course, all these parameters are learned from the training sample. In [11] is presented the analysis showing why the introduced parameters in RIONIDA and their proper selection are important factors for obtaining high performance of RIONIDA.

The learning process of these optimal parameters is discussed in Section 4.4. Algorithm 4 presents the RIONIDA algorithm for the testing phase. In the input of the algorithm, all metrics are given (used for computation of the final metric), but in the sg-rule, only metrics for symbolic attributes are used.

At the step of classification, a performance measure dedicated to imbalanced data does not appear in RIONIDA. Here, it is assumed that these parameters have been optimised for this chosen (by a user) measure.

RIONA (ONN) is obtained from RIONIDA after setting the threshold p at 0.5, the parameter s at 1.0 (-1.0), and the optimisation measure to Accuracy. Thus, RIONIDA is an extension of RIONA and ONN as well.

While developing the RIONIDA algorithm, preliminary experiments were conducted to analyse how specific performance measures for imbalanced data changed across the space of introduced parameters. Graphs showing the relationship between performance measures and parameter spaces (or subspaces) were constructed. The presence of distinct maximal points with values significantly higher than other points suggested that the introduced parameters could help find optimal parameter settings where specific performance measures approached their maximal values. The analysis presented below aims to show that it is indeed reasonable to introduce the discussed additional parameters and to search the space of them at the learning step. In the following subsections, four important

Algorithm 4: RIONIDA-classify.

Input: test example tst, training set $trnSet$, positive integer k, number $p \in [0,1]$, number $s \in [-1,1]$, family of metrics for attributes $\{\varrho_a\}_{a \in A}$
$\varrho = Agr(\{\varrho_a\}_{a \in A})$
$nS = N(tst, trnSet, k, \varrho)$
$supportSet(d_{min}) = \emptyset$
$supportSet(d_{maj}) = \emptyset$
Output: predicted decision for tst
for all $trn \in nS$ **do**
  $v = d(trn)$
  if $isConsistent\left(sg\text{-}rule\left(tst, trn, \{\varrho_a\}_{a \in A_{sym}}, s\right), nS\right)$ **then**
    $supportSet(v) = supportSet(v) \cup \{trn\}$
  end if
  $p_{current} = \frac{|supportSet(d_{min})|}{|nS|}$
  if $p_{current} \geq p$ **then**
    return d_{min}
  else
    return d_{maj}
  end if
end for

issues related to RIONIDA are analysed. They relate to (1) using (proper) performance measure for optimisation, (2) using (proper) neighbourhood size, (3) using (different) weights for decision classes, and (4) using consistency checking degree. The last three of them directly relate to the introduced additional parameters.

4.3.1. Selection of performance measure for optimisation

Normally, this measure is used during the evaluation of the learning algorithm at the testing stage. However, it is natural to make use of this measure in the optimisation of the learning algorithm relative to this measure (measure optimisation, for short) at the learning stage. In fact, we make use of it in the development of RIONIDA.

In RIONA, the Accuracy was used to evaluate this algorithm, and this performance measure was estimated at the learning stage. In RIONIDA, performance measures, more relevant for imbalanced data, e.g. F-measure, G-mean, AUC or other (e.g. defined by experts from confusion matrix), could be used. Currently, in the RIONIDA implementation, one can choose F-measure, G-mean or Accuracy at the learning stage. Primarily, we choose one out of two: F-measure or G-mean (performance measures relevant for imbalanced data).

4.3.2. Choice of the neighbourhood size

Using the neighbourhood of a test case (instead of the whole training set) of relevant size, it is both fast and effective (in classification). Analogously, we implement such an idea for RIONIDA (for performance measures more relevant for imbalanced data) and took 100 as a default bound for the neighbourhood size.

First, we discuss how parameter k affects the optimisation of performance measure for RIONIDA. To simplify the presentation, let us fix two parameters p and s of RIONIDA. A natural candidate for the default value of the parameter p is the percentage of the minority class in the training set, i.e. $\frac{|Class(d_{min})|}{|trnSet|}$. As the default value of parameter s, let us take $s = -0.1$ for which the sg-rule works exactly as for kNN (see Definition 3). In this way, we get an algorithm that we call Optimal Neighbourhood for Imbalanced Data Algorithm (ONIDA). Let us also assume that we want to optimise G-mean measure. The classification quality (as a function of parameters) was computed using the leave-one-out method applied to the whole data set.

Fig. 2 shows the dependency of G-mean measure on the parameter k for *glass* data set (for details about this data set and others mentioned below, see Table 2). For this data set, it can be observed that while increasing k beyond a certain small value (around 10), the G-mean measure is systematically falling down. It is clear from that graph that using a different setting of value k can produce classifiers with completely different quality. In the graph, we observe differences in G-mean of about 40%.

Similar results were observed in the performed experiments for F-measure, i.e. choosing the proper size of the neighbourhood (parameter k) can improve the classification quality for the ONIDA algorithm, the simplified version of RIONIDA.

4.3.3. Balancing sensitivity and specificity

In RIONA, it was assumed that the cardinality of decision classes is fairly evenly distributed. Conflict resolution for inducing decision was done in favour of the most-represented class in the neighbourhood of the test object.

In the case of imbalanced data, we assume that the minority class may be under-represented. To increase the chance of correct classification of objects from the minority class, objects from this class should be treated differently in comparison to those from the majority class.

For this purpose, we introduce a parameter p used to define how important the minority class is. This is, in a sense, analogous to changes made in the MODLEM-C algorithm [34] (in comparison to MODLEM [36]), where a fixed weight for examples from the minority class is assigned.

The parameter p of RIONIDA determines the minimum rate value of the number of objects (forming consistent sg-rules) from the minority class to the size of the whole neighbourhood for assigning the minority decision to the test object tst.

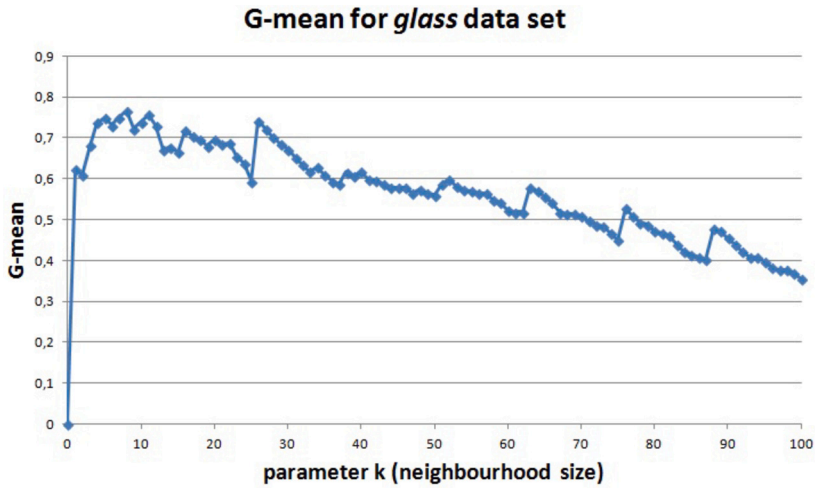

Fig. 2. G-mean measure for *glass* data set for the ONIDA algorithm as a function of parameter k (neighbourhood size).

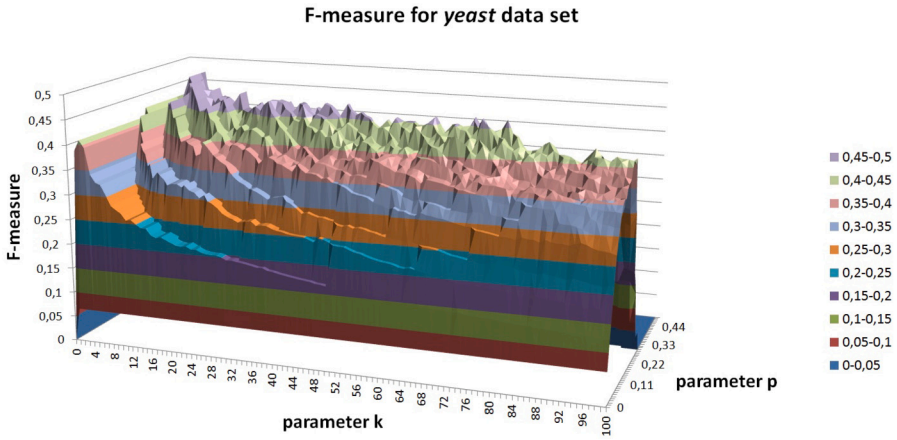

Fig. 3. Surface chart representing F-measure for RIONIDA for *yeast* data set as a function of parameters k and p with fixed $s = -0.1$.

The condition $p_{current} \geq p$ in Algorithm 4 is equivalent to assigning weights to minority and majority examples with values $1 - p$ and p, respectively.

Different values of this parameter give different weights for the minority class and the majority class. In consequence, this is related to different levels of sensitivity to the minority class, i.e. Sensitivity (and inversely, sensitivity to the majority class, i.e. Specificity).

From the perspective of optimisation for G-mean (or F-measure), searching for optimal p value corresponds to searching for an appropriate point on the ROC curve (or Precision-Recall curve).

We want to observe how different pairs of the parameters p and k can affect the values of the performance measure we are interested in. Thus, we fixed the parameter s at -0.1. Fig. 3 shows the dependency of F-measure on both parameters k and p for the exemplary data set. The maximum F-measure value is obtained for $k = 12$ and $p \in [0.26, 0.33]$.

For G-mean, the dependency is different, and the maximum G-mean value (for the same data set) is obtained for $k = 48$ and $p \in [0.03, 0.04]$.

These considerations support a hypothesis that it is worth finding the optimal value of the parameter p according to the given performance measure we want to optimise. This optimal value of the parameter p can be different for different performance measures.

4.3.4. *Choice of scaling factor in the sg-rule*

In RIONA, only those objects from the neighbourhood are counted that support the consistent rule generated by RIONA. It was reported that ONN (a modification of RIONA) that takes into account all objects from the neighbourhood sometimes achieves better quality than RIONA. Thus, one can try to learn from the training sample which algorithm to apply for a specific data set. Even more, one can also introduce a smooth transition between these two situations. This is done by introducing the parameter s (see Definition 3).

The value of s corresponds to the degree of consistency rule detection. The value $s = 1$ corresponds to the situation as in RIONA. In this sense, RIONIDA is an extension of this algorithm. The value $s = -0.1$ corresponds to the ONN method, i.e. we do not check

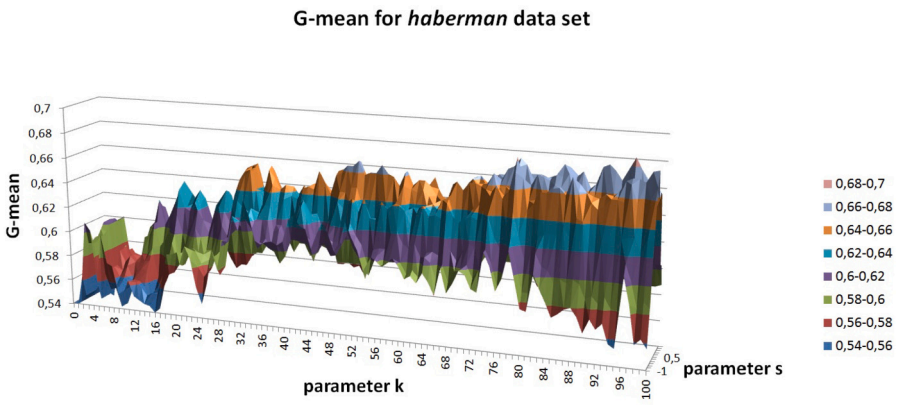

Fig. 4. Surface chart representing G-mean measure (scaled from 0.58) for RIONIDA for *haberman* data set as a function of parameters k and s with fixed $p = 0.22$.

the consistency of examples. Intermediate values, i.e. $0 < s < 1$, correspond to the situations between the ONN algorithm and the rule-based algorithm.

Fig. 4 shows the dependency of G-mean measure on both parameters k and s for *haberman* data set. We have fixed here parameter $p = 0.22$ (close to the percentage of the minority class in the whole data set; for this value of p the maximum value of G-mean was obtained over the set of values for the parameters k, p, s). It is visible that for almost all values of the parameter k the maximum value of G-mean is obtained for some value of the parameter s between 0 and 1, near 0.5.

If we fix $k = 96$ and $p = 0.22$, the maximum value of G-mean is obtained for $s = 0.5$, and the difference between maximal (for $s = 0.5$) and minimal (for $s = 1.0$, i.e. for rule-based classifier) value is approximately 10%. The difference between the maximum G-mean value (for $s = 0.5$) and G-mean value for $s = -0.1$ (i.e. for the kNN like classifier) is approximately 8%.

Also, we considered the maximal G-mean over all values of k and p for RIONIDA for *haberman* data set as a function of parameter s. The difference between the maximum G-mean value (for $s = 0.5$) and (i) G-mean value for $s = 1$ (i.e. for rule-based classifier) is approximately 1.8%; (ii) G-mean value for $s = -0.1$ (i.e. for kNN like classifier) is approximately 2.1%. Still, these differences show a possibility for significant improvements of both rule-based classifier and kNN-based classifier by using a classifier 'between' these two.

Generally, we have such a division: for some data sets, the maximum value of the optimised measure is reached for $s = -0.1$, i.e. for methods of the kNN type. For another part of the data sets, the maximum value is reached for $s = 1.0$, which corresponds to the application of rules. In turn, for a part of data sets, the maximum value is reached for $s \in (0, 1)$, which corresponds to the application of both of these: the rule-based method to some extent and kNN-based method to some extent.

4.4. Estimating the optimal values of parameters for RIONIDA

The considerations in Subsections 4.3.2-4.3.4 show that the performance of RIONIDA can significantly depend on the chosen values of the parameters k, p, s. The optimal values of these parameters depend on the analysed data set and the selected optimisation measure. Therefore, it is essential to find the optimal values relative to the optimisation measure specified by a user. It should be noted that the domains of the parameters k, p, s (maximal admissible sets for these parameters) are as follows: $K_{max} = \{1, 2, \ldots, |trnSet|\}$, $P_{max} = [0, 1]$, $S_{max} = \{-0.1\} \cup [0, 1]$. We would like to search for the optimal triple values in the Cartesian product of these sets. From the algorithmic point of view, one should restrict the search to some finite subsets of these sets. By default, we use sets $K = \{1, 2, \ldots, 100\}$, $P = \{0.01, 0.02, \ldots 0.5\}$, $S = \{-0.1, 0.0, 0.1, \ldots, 1.0\}$ with a size of 100, 50 and 12 respectively.

Analogously as in the case of RIONA, to construct an efficient algorithm, one should take into account the following question: For given finite sets K, P, S, how to learn the optimal triple values efficiently from $K \times P \times S$?

In [26], the estimation of the parameter p was done for some specific situations. Also, it was experimentally shown that for many data sets, the estimation is very close to the optimal value of p.

4.4.1. Efficient learning of the optimal values of parameters for RIONIDA

Here, we discuss the algorithm for estimation of the optimal values of the parameters k, p, and s for RIONIDA. This can be done in an analogous way to searching for the optimal value of k in the case of RIONA. The leave-one-out method is used on the given training set to estimate the value of the performance measure (chosen by a user) for different values of $(k, p, s) \in K \times P \times S$ and the triple values of k, p, s for which the estimation of the measure value is the greatest is selected. The direct calculations require repeating leave-one-out estimation $|K| \cdot |P| \cdot |S|$ times. However, using the dynamic programming technique (and the idea analogous to that in RIONA), we emulate this process in time much lower than this. Below we present Algorithm 6 implementing this idea.

For a training example trn, the function $getClassificationMatrix$ (see Algorithm 5) finds k_{max} examples from $trnSet \setminus \{trn\}$ nearest to the example trn and sorts them according to the distance $\varrho(trn, \cdot)$ from the trn object.

Next, for any example nn_k from the selected neighbourhood and any $s \in S$, the sg-rule is built on trn (treated as a testing object) and nn_k (treated as a training object), i.e. the rule $sg\text{-}rule\left(trn, nn_k, \{o_a\}_{a \in A_{sym}}, s\right)$. The algorithm checks the consistency of this sg-rule

Algorithm 5: getClassificationMatrix(trn, $trnSet$, K, P, S, $\{\varrho_a\}_{a\in A}$).

Input: currently considered example $trn \in trnSet$, training set $trnSet$,
K, P, S – sets of admissible values for parameters k, p, s, respectively,
family of metrics for attributes $\{\varrho_a\}_{a\in A}$
Output: 3 dimensional matrix (for different triple values $(k,p,s) \in K \times P \times S$) of leave-one-out classification for trn
$k_{max} = |K|$ (we assume that K is the set of consequent natural numbers)
$\varrho = Agr(\{\varrho_a\}_{a\in A})$
$N = N(trn, trnSet \setminus \{trn\}, k_{max}, \varrho)$
vector $(nn_1, \ldots, nn_{|N|}) = (N$ sorted according to the distance $\varrho(trn, \cdot))$
for $k = 1$ **to** $|N|$ **do**
  for all $s \in S$ **do**
    $nn_k.isConsistentOnLevel[s] = isConsistent(sg\text{-}rule\left(trn, nn_k, \{\varrho_a\}_{a\in A_{sym}}, s\right), N)$
  end for
end for
for all $s \in S$ **do**
  $decStrength[d_{min}] = 0$
  $decStrength[d_{maj}] = 0$
  for $k = 1$ **to** $|N|$ **do**
    if $nn_k.isConsistentOnLevel[s]$ **then**
      $v = d(nn_k)$
      $decStrength[v] = decStrength[v] + 1$
    end if
    $p = \frac{decStrength[d_{min}]}{decStrength[d_{min}]+decStrength[d_{maj}]}$
    for all $p_{current} \in P$ **do**
      $currentDec = d_{min}$
      if $p_{current} > p$ **then**
        $currentDec = d_{maj}$
      end if
      $M[k,p,s] = currentDec$
    end for
  end for
end for
return M

Algorithm 6: findOptimalParams3D($trnSet$, K, P, S, $optMeasure$, $\{\varrho_a\}_{a\in A}$).

Input: training set $trnSet$,
K, P, S – sets of admissible values for parameters k, p, s, respectively,
optimisation measure $optMeasure$ from {F-measure, G-mean, Accuracy},
family of metrics for attributes $\{\varrho_a\}_{a\in A}$
Output: triple of the optimal values of parameters k, p, s
for all $trn \in trnSet$ **do**
  $M_{trn} = getClassificationMatrix(trn, trnSet, K, P, S, \{\varrho_a\}_{a\in A})$
end for
fill $estimatedConfusionMatrix$ with values 0
for all $(k,p,s) \in K \times P \times S$ **do**
  for all $trn \in trnSet$ **do**
    $realDec = d(trn)$
    $classifierDec = M_{trn}[k,p,s]$
    $estimatedConfusionMatrix[k,p,s][realDec, classifierDec]{+}{+}$
  end for
  count $estimatedMeasure[k,p,s]$ from $estimatedConfusionMatrix[k,p,s]$ based on $optMeasure$
end for
return $\arg\max_{(k,p,s)\in K\times P\times S} estimatedMeasure[k,p,s]$

with the objects from the neighbourhood for different levels of $s \in S$ and stores this information in the entry corresponding to s of the array assigned to the object nn_k.

Next, it calculates the matrix of decisions that the RIONIDA classifier would return for different triple values $(k,p,s) \in K \times P \times S$ and this matrix is returned as a result.

Algorithm $findOptimalParams3D$ (see Algorithm 6) calls the function $getClassificationMatrix(\ldots)$ for every training object. Next, it creates a matrix with the confusion matrix as its entry for each triple $(k,p,s) \in K \times P \times S$. The entry of this matrix corresponding to the index defined by the values of the parameters k,p,s consists of the confusion matrix consisting of information for leave-one-out classification for these values of the parameters k,p,s over all training examples (excluding the considered one). Any confusion matrix (in the matrix of confusion matrices $estimatedConfusionMatrix$) is transformed into one value calculated using the selected optimisation measure $optMeasure$ (and stored in the matrix $estimatedMeasure$). Finally, it selects the triple of the optimal values of the parameters k, p, s for which the global estimation of the chosen optimisation measure is maximal.

This algorithm is analogous to the algorithm learning of the optimal parameter k in RIONA [10]. In this algorithm, the triple of the optimal values of the parameters k, p, s, rather than only one value of the parameter k is returned. Moreover, the optimal parameters relative to the given optimisation measure instead of the Accuracy measure are returned.

The algorithm $findOptimalParams3D$ has arguments K, P, S specifying the sets of admissible values for the parameters k, p, s, respectively. We assume that $K = \{1, 2, \ldots, k_{max}\}$, i.e. the admissible values of the parameter k are consecutive natural numbers.

Another argument of the algorithm is the optimisation measure $optMeasure$. In the current implementation, F-measure, G-mean or Accuracy can be substituted here as the value of this argument. However, from the description of the algorithm, it is clear that any optimisation measure, which is the function of the confusion matrix, could also be used.

4.5. Time and space complexity of RIONIDA

In this section, we analyse the time and space complexity of RIONIDA. Moreover, we show how both presented complexity bounds can be improved for the learning phase.

4.5.1. Time complexity of RIONIDA for the testing phase

The analysis of the RIONIDA algorithm in the testing phase is very similar to the RIONA algorithm [10]. In any run of the RIONIDA algorithm, two phases can be distinguished. In the first phase, training examples from the neighbourhood N are selected. In the second phase, the algorithm checks consistency among them. The time complexity of RIONIDA is the same as for the RIONA algorithm. Under the assumption made at the beginning of this subsection (on the size of the neighbourhood N), the time complexity of RIONIDA is $O(m(n + k^2))$ for a single test object, where $n = |trnSet|$, $m = |A|$.

In the case when k is treated as a constant (or $k < \sqrt{n}$), the time complexity of the testing phase (for single test object) for RIONIDA is $O(mn)$.

4.5.2. Time complexity of RIONIDA for the learning phase

The analysis of time complexity of the learning phase for RIONIDA is in many aspects analogous to RIONA [10].

Theorem 1. *The time complexity of the learning phase of RIONIDA is $O(mn^2 + n|S| \cdot k_{max} \cdot (mk_{max} + |P|))$, where $n = |trnSet|$, $m = |A|$, $k_{max} = |K|$ is the parameter used to define the maximal size of the neighbourhood to be analysed, P, S are sets of admissible values of the parameters p, s, respectively.*

Proof. For any training object, in the run of the learning algorithm (see lines 3-5 of Algorithm 6) one can distinguish four phases (realised by Algorithm 5).

In the first phase, training examples from the neighbourhood N are selected, i.e. k_{max} nearest objects to the considered training example among n objects, where $n = |trnSet|$. It can be done in the linear time relative to n (see e.g. [20, pp.189-192]). Taking into account that for any object all attributes should be examined, time complexity of this phase is $O(mn)$, where $m = |A|$.

In the second phase, all selected objects from the neighbourhood N are sorted. Computing distances for objects from N takes $O(m|N|)$ steps (once for every object from N). Sorting (using computed distances) can be done in $O(|N| \log |N|)$ steps. Thus, this phase takes $O(m|N| + |N| \log |N|)$ steps.

In the third phase, the algorithm checks consistency (and marks it) among selected objects for different values of the parameter s from the set S (see lines 7-11 of Algorithm 5). It takes $O(|S| \cdot m \cdot |N|^2)$ steps.

From the assumption on the bound of the neighbourhood N, the second and third phases altogether take $O(|S| \cdot mk_{max}^2)$ steps.

In the fourth phase, the algorithm fills the classification matrix M on the basis of the marked consistency (see lines 12-29 of Algorithm 5). It takes $|S| \cdot |K| \cdot |P|$ steps.

Thus, the method $getClassificationMatrix$ takes $O(mn + |S| \cdot mk_{max}^2 + |S| \cdot |K| \cdot |P|)) = O(mn + |S| \cdot k_{max}(mk_{max} + |P|))$ steps. This method is executed for each training example. Thus, the time complexity of *foreach* loop within lines 3-5 of Algorithm 6 is $O(mn^2 + n|S| \cdot k_{max} \cdot (mk_{max} + |P|))$.

Finally, for the whole training set, the algorithm computes the leave-one-out confusion matrix for each triple $(k, p, s) \in K \times P \times S$ (see lines 6-14 of Algorithm 6). This takes $O(nk_{max} \cdot |P| \cdot |S|)$ steps.

Summing up, the time complexity of the learning algorithm is $O(mn^2 + n|S| \cdot k_{max} \cdot (mk_{max} + |P|))$. □

If we assume that $|P| \leq mk_{max}$ (which is true in our primary experiments), then the time complexity of the learning algorithm is $O(m(n^2 + n|S| \cdot k_{max}^2))$.

4.5.3. Space complexity of RIONIDA for the learning phase

Fact 2. *The space complexity of the learning phase for RIONIDA is $O(n \cdot |K| \cdot |P| \cdot |S|)$, where $n = |trnSet|$, K, P, S are sets of admissible values of the parameters k, p, s, respectively.*

Proof. The space complexity of the learning phase for RIONIDA is mainly related to allocating matrices for all training examples of the size $|K| \cdot |P| \cdot |S|$ (see lines 3-5 of Algorithm 6). Thus, allocated space is of the size $O(n \cdot |K| \cdot |P| \cdot |S|)$, where $n = |trnSet|$. For the matrices *estimatedConfusionMatrix* and *estimatedMeasure* it is necessary to allocate space $O(|K| \cdot |P| \cdot |S|)$. Thus, the overall space complexity of the learning phase for RIONIDA is $O(n \cdot |K| \cdot |P| \cdot |S|)$. □

Table 2
Description of data sets used in experiments.

Data set name	Identifier	No of examples	No of conditional attributes (numerical, nominal)	No of original classes	Classes for binary classification task (minority class, majority class)	Minority class (in %)
Abalone	abalone	4177	8 (7, 1)	29	(1-4 and 16-29, others)	8.02
Balance Scale	balance-scale	625	4 (0, 4)	3	(B = balanced, others)	7.84
Breast Cancer	breast-cancer	286	9 (0, 9)	2	(recurrence-events, no-recurrence-events)	29.72
Breast Cancer Wisconsin (Original)	breast-w	699	9 (9, 0)	2	(malignant, benign)	34.48
Car Evaluation	car	1728	6 (0, 6)	4	(good, others)	3.99
Heart Disease (Cleveland)	cleveland	303	13 (6, 7)	5	(3, others)	11.55
Statlog (German Credit Data)	credit-g	1000	20 (7, 13)	2	(bad, good)	30.00
Ecoli	ecoli	336	7 (7, 0)	8	(imU, others)	10.42
Glass Identification	glass	214	9 (9, 0)	7	(3 = vehicle_windows_f_p, others)	7.94
Haberman's Survival	haberman	306	3 (3, 0)	2	(1 = the patient survived, 2 = died)	26.47
Hepatitis	hepatitis	155	19 (6, 13)	2	(1 = die, 2 = live)	20.65
Ionosphere	ionosphere	351	34 (34, 0)	2	(bad, good)	35.90
Microcalcifications in Mammography	mammography	11183	6 (6, 0)	2	(1 = abnormal pixels, 0 = normal pixels)	2.33
Thyroid Disease	new-thyroid	215	5 (5, 0)	3	(2 = hyper, others)	16.28
Nursery	nursery	12960	8 (0, 8)	5	(very_recom, others)	2.53
Pima Indians Diabetes	pima	768	8 (8, 0)	2	(1 = tested positive for diabetes, 0)	34.90
Post-Operative Patient	postoperative	90	8 (0, 8)	3	(S = patient prepared to go home, others)	26.67
Blood Transfusion Service Center	transfusion	748	4 (4, 0)	2	(1 = donated blood, 0 = not donated)	23.80
Statlog (Vehicle Silhouettes)	vehicle	846	18 (18, 0)	4	(van, others)	23.52
Yeast	yeast	1484	8 (8, 0)	10	(ME2, others)	3.44

5. Experiments and main results

We compare our RIONIDA algorithm with other state-of-the-art algorithms on several benchmarks. The experiments were designed to be reproducible.

To evaluate learning algorithms, (and also generated by them classifiers), we use two performance measures: F-measure and G-mean.[5]

For estimation of the mentioned performance measure, we use 10 times repeated 10-fold stratified cross-validation. Partial results of each 10-fold stratified cross-validation are micro-averaged. As the final estimation of the desired measure, the average of ten repetitions of this procedure is used. For all compared learning algorithms, the same splits in the cross-validation process are used. It can be thought that the estimation is done in parallel for all learning algorithms. In practice, we simply use the same random seed (used for random partitions of data sets) for all learning algorithms in the process of estimating the chosen measure. This guarantees the reproducibility of experiments.

The overwhelming majority of experiments were conducted using PowerShell scripts on Windows 10, scripts in Java, and R environment on a computer equipped with Intel(R) Core(TM) i7 CPU 870 @ 2.93GHz and 8GB RAM.[6]

Data sets used in experiments are based mainly on UCI Machine Learning Repository [29].[7] Data sets containing originally more than two classes were transformed into the binary classification task by choosing one small class or joining several small classes into one (minority) class; other classes were joined into another (majority) class.

Table 2 presents all data sets used in the experiments. This choice of data sets seems to create a relevant base for experiments since we have selected 20 fairly diverse imbalanced data sets considering the aspects described below.

- The size of data sets is varied (from 90 to 11180 examples in total).
- The percentage of the minority class is varied (from 2.33% to 35.90%, i.e. imbalance ratio is between around 2 and 42).
- The types of attributes are also varied (either only numerical, either only symbolic or mixed numerical and symbolic).
- The data set difficulty, in terms of types of examples (safe, rare, borderline or outlier) [31] is also varied. In [31], most of the data sets that we used in our experiments were inspected. For example, out of the data sets inspected in [31] and occurring in our experiments, the most difficult data sets are: (sorted in order from the 'most difficult' to 'easier ones') *balance-scale*, *yeast*, *transfusion*, *postoperative*, *abalone*, *glass*, *cleveland*. These data sets contain a small number (or even none) of safe examples, more than 25% of outlier examples and a relatively high number of borderline or rare examples. For example, *balance-scale* data set contains no safe example and no borderline example, 8.16% of rare examples and 91.84% of outlier examples; *cleveland* data set

[5] We have not used AUC measure because of (i) the criticism about it (see e.g. [27,28]); (ii) BRACID, one of the important learning algorithms that we wanted to compare with, does not return probabilities for the two decision classes (only the deterministic decision is returned).

[6] Some additional experiments were conducted using PowerShell scripts on Windows 10, scripts in Java, and R environment on a computer equipped with Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz and 8GB RAM.

[7] Only *mammography* data set is not publicly available and was supported by Nitesh Chawla [30].

contains no safe example, 31.43% borderline examples, 17.14% of rare examples, and 51.43% of outlier examples (for details and information about other data sets, see [31]).

- There are both consistent and inconsistent data sets. There are data sets with and without missing values.

We use the Friedman statistical test (see [32,33]) for comparing multiple learning algorithms on multiple data sets. If this test passes, then we use the Finner post-hoc test.[8] For all tests, we use the significance level $\alpha = 5\%$. The statistical analysis was done using the R Project for Statistical Computing, commonly known as the R (see https://www.R-project.org).

Generally, one of the aims of performed experiments was to compare the new RIONIDA algorithm with some other state-of-the-art learning algorithms. Some of them are specially designed for the classification of imbalanced data, and some are not. One can also use state-of-the-art learning algorithms developed for balanced data and apply them to the results of sampling methods (filters) dedicated to imbalanced data. We use two types of well-known filters. Below we describe all learning algorithms and filters used in the experiments.

Algorithms developed for the analysis of imbalanced data together with their short descriptions are as follows.

BRACID [5] analogously to RISE uses an integrated representation of rules and single instances. It comprehensively addresses the issues associated with imbalanced data. It uses the strategy of bottom-up induction of rules from single examples with the specific strategy of generalisation. A conflict resolution is based on the supports of the nearest rules to the test example.

MODLEM-C [34] is an extension of MODLEM (see below) with the possibility of strengthening Sensitivity. The rule strength is multiplied for all rules describing the minority class by the same real number called the *strength multiplier* given as a parameter.

The remaining algorithms (generally dedicated to balanced data) used in the experiments are: kNN [35] (it can select the relevant value of k based on cross-validation); MODLEM [36]; J48 (decision tree learning algorithm) [37]; PART [38]; RIPPER [39]; RISE [15]; RIONA.

We used three strategies for comparing RIONIDA with other algorithms. These strategies are related to three levels of increasing challenge for RIONIDA in relation to the other algorithms.

The first strategy, which we refer to as the *def* strategy, uses the default option settings for the algorithms and the `SMOTE+ENN` filter, i.e. `SMOTE` is applied first, followed by Edited Nearest Neighbor (`ENN`). `SMOTE` is an over-sampling method, which creates new synthetic examples for all minority class examples (for more details, see [30]). `ENN` is an under-sampling method, which discards those majority examples which are close to the minority class (for more details, see [40]). The motivation for selecting `SMOTE+ENN` comes from [41], which showed that this filters combination provides, in practice, very good performance compared to other filters combinations for data sets with a small number of positive examples. We use this filter for algorithms which are not dedicated to imbalanced data (for algorithms dedicated to imbalanced data no filters are used). *Def* strategy is commonly used in the literature for comparative experiments (default use of the algorithm is applied; possibly preceded with some fixed filter).

Let us note that the parameter optimisation is built into the RIONIDA algorithm, and one can ask to perform a comparison with mentioned algorithms using a similar extent of parameter optimisation. Thus, additionally, we performed experiments with other two levels of increasing challenge for RIONIDA in relation to the other algorithms used in the comparison.

In the two latter strategies, we allow different than default option settings of the algorithms and different filters (excluding RIONIDA[9] and BRACID). For each learning algorithm, we allow the combination of option settings from the fixed, predefined set. We also allow the use of one of three possibilities of filtering (`SMOTE`, `SMOTE+ENN`, or no filtering). Combination of both (1) option settings for algorithm and (2) one of the filters we call *algorithm configuration* (or just *configuration* if the context is clear). For example, in the mentioned *def* strategy, each algorithm configuration is fixed (default options and default mentioned above filtering).

In the second strategy, *opt* strategy, for each algorithm (excluding RIONIDA and BRACID), we used globally 'optimal' (for the used data sets) algorithm configuration. The 'optimal' means the one with the lowest (optimal) rank in the group of different configurations (for details see [11]). The intuition is that we want to select the algorithm configuration, which will be the most competitive in the context of the Friedman statistical test used at the end of the comparative process (since the Friedman statistical test uses average ranks of the learning algorithms). In particular, if all the learning algorithms were set by chance with other options and filters than the default ones, then one could expect that the obtained results were lower (for other algorithms than RIONIDA) than the results used in this strategy. If, for this strategy, RIONIDA could be shown to be statistically better than some algorithms, then this should be perceived as a stronger (than with *def* strategy) result in favour of the RIONIDA algorithm.

In the third, *max* strategy (the most competitive of the three presented), we emulated learning of optimal algorithm configurations. For each algorithm (excluding RIONIDA and BRACID) and for each data set, we simply selected the maximal score of performance measure for many of its configurations and used such maximal value (favouring other algorithms used in our comparison!) for the final statistical comparison in the Friedman statistical test.

Let us assume that for each algorithm used in the experiments, one constructs a meta-learning algorithm that learns the optimal algorithm configuration for a given training sample (using some validation scheme). For each algorithm, the *max* strategy provides an (upper) approximation of the scores of such meta-learning algorithm. In particular, if all learning algorithms were supported with the possibility of learning of the optimal algorithm configurations (using only the training data), one could expect that the obtained scores were lower (for other algorithms than RIONIDA) than the presented ones in the comparisons for the *max* strategy. If for this

[8] For some of our comparisons also conservative Nemenyi post-hoc test could show analogous statistical conclusions as presented further in the article (see [11]).

[9] This makes that the two latter discussed strategies favour other algorithms over RIONIDA (and BRACID). BRACID is also excluded just because it has no available options, and as dedicated for imbalanced data, no filtering is used.

Table 3
The values of G-mean (in %) for different algorithms and data sets for the *def* strategy. The RIONIDA algorithm was set to optimise the G-mean measure (i.e. $\text{RIONIDA}_{\text{G}}$ was used). For each data set, the best-obtained score is shown in bold. Also, for illustration, for five algorithms on the right (including $\text{RIONIDA}_{\text{G}}$), ranks for these algorithms and different data sets are shown (in parentheses). At the bottom are shown: (i) average rank for each algorithm, (ii) important outcomes of the Friedman statistical test (Friedman statistic, degrees of freedom, and p-value), and (iii) adjusted p-values (APV) with the Finner post-hoc test using RIONIDA as the control algorithm.

	The vectors of representative scores for G-mean chosen as perf. measure of our interest for the *def* strategy									
Data set	kNN	PART	J48	RIPPER	RISE	MODLEM	MODLEM-C	RIONA	BRACID	$\text{RIONIDA}_{\text{G}}$
abalone	59.39	70.06	70.36	**73.05**	60.03	65.54 (6)	55.06 (10)	59.91 (8)	65.80 (5)	67.94 (4)
balance-scale	56.97	47.03	22.70	13.58	40.97	12.90 (9)	2.78 (10)	33.26 (6)	58.68 (2)	**76.98 (1)**
breast-cancer	56.64	53.99	54.23	53.53	58.26	56.04 (7)	58.34 (2)	56.92 (5)	58.17 (4)	**64.98 (1)**
breast-w	97.36	96.28	95.79	95.99	96.89	96.16 (7)	94.89 (10)	**97.81 (1)**	96.91 (4)	97.53 (2)
car	83.23	86.68	87.40	80.08	75.89	82.51 (7)	89.23 (2)	80.37 (8)	87.47 (3)	**96.74 (1)**
cleveland	63.83	63.77	66.48	69.68	59.43	63.34 (7)	35.61 (10)	65.27 (4)	62.89 (8)	**76.38 (1)**
credit-g	65.66	66.30	66.17	65.59	65.27	65.57 (8)	66.78 (2)	66.35 (3)	62.27 (10)	**69.90 (1)**
ecoli	86.82	84.82	84.11	86.36	85.59	84.15 (8)	67.65 (10)	86.68 (3)	84.42 (7)	**88.82 (1)**
glass	62.75	64.30	67.89	57.00	54.69	63.16 (5)	47.64 (9)	66.80 (3)	39.90 (10)	**69.26 (1)**
haberman	59.76	61.64	62.41	61.90	60.35	62.64 (2)	57.10 (10)	59.85 (7)	59.55 (9)	**65.40 (1)**
hepatitis	74.94	67.56	66.78	65.82	71.16	71.71 (5)	67.55 (8)	73.46 (4)	77.11 (2)	**79.00 (1)**
ionosphere	89.91	86.88	85.64	84.27	**91.91**	85.93 (8)	89.55 (6)	90.37 (4)	91.42 (2)	90.89 (3)
mammography	73.48	72.59	71.73	73.37	73.18	70.68 (9)	68.74 (10)	74.17 (3)	85.41 (2)	**89.70 (1)**
new-thyroid	98.71	95.13	95.05	95.16	97.73	94.36 (9)	92.92 (10)	**98.93 (1.5)**	98.69 (4)	**98.93 (1.5)**
nursery	89.99	97.12	87.04	84.43	83.82	97.05 (4)	99.80 (2)	88.73 (7)	96.58 (5)	**99.90 (1)**
pima	66.75	67.07	67.47	68.42	67.99	65.41 (10)	69.96 (3)	66.54 (9)	71.28 (2)	**72.87 (1)**
postoperative	38.32	36.03	37.33	33.91	36.84	34.75 (8)	40.21 (3)	34.02 (9)	42.49 (2)	**43.66 (1)**
transfusion	62.76	61.89	63.22	64.38	63.11	62.31 (8)	57.57 (10)	63.34 (4)	64.39 (2)	**67.64 (1)**
vehicle	93.27	93.51	93.03	93.06	92.59	93.67 (5)	**95.45 (1)**	94.47 (3)	93.82 (4)	95.10 (2)
yeast	74.34	71.43	70.08	73.60	68.78	64.55 (9)	46.95 (10)	75.92 (2)	72.38 (5)	**84.95 (1)**
average rank	5.2	5.9	6.3	6.45	6.5	7.05	6.9	4.725	4.6	**1.375**
Friedman test	Friedman's chi-squared = 55.814, df = 9, p-value = $8.52 \cdot 10^{-9}$									
APV Finner	0.00008	$< 10^{-5}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-7}$	$< 10^{-7}$	0.00053	0.00076	control

strategy, RIONIDA could be shown to be statistically better than some algorithms, this should be perceived as a very strong result in favour of the RIONIDA algorithm.

For more details about these three strategies, in particular, used fixed, predefined algorithm configurations, see [11].

5.1. Comparison of algorithms for G-mean

In this subsection, we assume that the performance measure we are interested in is G-mean. Thus, the particular parameter of RIONIDA is set to optimise G-mean. The algorithm with this setting is called $\text{RIONIDA}_{\text{G}}$.

For each learning algorithm, the representative scores of G-mean for all used data sets were computed.

5.1.1. Def *strategy for G-mean*

In this step, we compare algorithms using their default algorithm configurations. In Table 3, for each learning algorithm, the representative scores of G-mean for all used data sets are given. The RIONIDA algorithm was set to optimise the G-mean measure, i.e. $\text{RIONIDA}_{\text{G}}$ was used; hence, $\text{RIONIDA}_{\text{G}}$ appears in the table instead of RIONIDA.

For considered 20 data sets, 15 times RIONIDA wins with all other algorithms (achieves the best score; rank equal to 1), and once (for *new-thyroid*) its score is equal to the other algorithm (namely, RIONA) with the best score (in this case, RIONIDA has a rank equal to 1.5).

The average of all ranks for RIONIDA gives the result of 1.375, which is the best outcome. The difference between the average rank of RIONIDA and the second-lowest average rank (4.6 for BRACID) is relatively high (3.225).

The Friedman statistic was computed, and the obtained p-value is much smaller than $\alpha = 0.05$. Thus, we can perform a post-hoc test.

Finner statistical test was used to compare all learning algorithms with the $\text{RIONIDA}_{\text{G}}$ algorithm set as the control one. For all learning algorithms used in comparisons with $\text{RIONIDA}_{\text{G}}$, the corresponding p-value is (much) smaller than $\alpha = 0.05$ (for details, see [11]). Thus, we can (confidently) reject all the null hypotheses corresponding to the algorithms used in the comparison (at 0.05 level of significance). In other words, $\text{RIONIDA}_{\text{G}}$ is significantly better than any other learning algorithm (with default algorithm and filter settings) relative to the G-mean performance measure.

5.1.2. Opt *strategy for G-mean*

For this strategy, the average ranks for algorithms are as follows: 5.1 (kNN); 5.825 (PART); 6.6 (J48); 6.55 (RIPPER); 6.5 (RISE); 6.25 (MODLEM); 6.975 (MODLEM-C); 4.925 (RIONA); 4.8 (BRACID); 1.475 (RIONIDA). Thus, again RIONIDA achieved the best outcome. For a detailed table with the representative scores, see [11]. The p-value of the Friedman statistical test is $7.235 \cdot 10^{-8}$ (much

Table 4
The values of G-mean (in %) for different algorithms and data sets for the *max* strategy (additionally, for RIONIDA ranks are shown in parentheses). The RIONIDA algorithm was set to optimise the G-mean measure (i.e. RIONIDA_G was used). It should be noted that for both RIONIDA_G and BRACID, one default algorithm configuration was used (i.e. no filter and default parameters of the algorithm were used) – their scores are the same as in the *def* strategy. For the other learning algorithms, the vector of representative scores was generated using the *max* strategy. For each data set, the best-obtained score is shown in bold. At the bottom are shown: (i) average rank for each algorithm, (ii) important outcomes of the Friedman statistical test (Friedman statistic, degrees of freedom, and p-value), and (iii) adjusted p-values (APV) with the Finner post-hoc test using RIONIDA as the control algorithm.

	The vectors of representative scores for G-mean chosen as perf. measure of our interest for the *max* str.									
Data set	kNN	PART	J48	RIPPER	RISE	MODLEM	MODLEM-C	RIONA	BRACID	RIONIDA_G
abalone	63.46	71.78	70.85	**73.05**	60.03	65.54	55.06	59.91	65.80	67.94 (4)
balance-scale	65.34	56.74	23.37	15.45	40.97	12.90	2.78	33.26	58.68	**76.98 (1)**
breast-cancer	59.22	55.50	55.35	57.94	58.49	58.87	58.34	60.97	58.17	**64.98 (1)**
breast-w	97.46	96.35	95.79	96.21	97.02	96.16	94.89	**97.81**	96.91	97.53 (2)
car	86.23	95.45	90.23	80.81	76.90	89.09	89.24	85.29	87.47	**96.74 (1)**
cleveland	75.46	67.19	66.49	69.68	59.43	63.34	35.61	65.27	62.89	**76.38 (1)**
credit-g	65.66	66.78	66.17	65.59	65.27	65.65	66.87	66.35	62.27	**69.90 (1)**
ecoli	88.35	86.44	84.45	86.44	85.59	84.15	67.65	86.68	84.42	**88.82 (1)**
glass	68.03	66.66	**69.44**	57.99	54.69	63.82	47.64	67.69	39.90	69.26 (2)
haberman	59.88	61.75	63.80	62.48	60.35	62.64	57.10	61.00	59.55	**65.40 (1)**
hepatitis	**80.09**	69.80	70.02	68.07	71.16	73.42	68.06	73.46	77.11	79.00 (2)
ionosphere	92.41	89.01	87.97	88.28	**92.82**	88.72	89.60	90.62	91.42	90.89 (4)
mammography	88.28	85.04	83.41	84.51	84.57	79.96	70.60	74.17	85.41	**89.70 (1)**
new-thyroid	**98.94**	95.92	95.15	96.03	97.73	95.33	92.98	98.93	98.69	98.93 (2.5)
nursery	91.52	**99.96**	95.13	85.74	95.59	99.80	99.80	99.56	96.58	99.90 (2)
pima	71.48	69.26	70.34	70.03	68.70	69.45	70.83	67.13	71.28	**72.87 (1)**
postoperative	39.48	37.20	37.87	34.24	36.84	34.75	40.21	34.02	42.49	**43.66 (1)**
transfusion	62.90	62.62	64.93	64.38	63.11	62.31	58.84	63.34	64.39	**67.64 (1)**
vehicle	94.21	94.05	93.14	93.94	92.59	95.24	**95.55**	95.18	93.82	95.10 (4)
yeast	79.83	72.76	70.46	73.62	68.78	64.55	46.95	75.92	72.38	**84.95 (1)**
average rank	3.9	5.225	6.05	6.475	6.75	6.575	7.425	5.275	5.6	**1.725**
Friedman test	Friedman's chi-squared = 53.725, df = 9, p-value = $2.13 \cdot 10^{-8}$									
APV Finner	0.02310	0.00029	0.00001	$< 10^{-5}$	$< 10^{-6}$	$< 10^{-5}$	$< 10^{-7}$	0.00027	0.00008	control

less than 0.05). The adjusted p-values for all the algorithms for the post-hoc Finner statistical test (with the RIONIDA_G algorithm set as the control one) are (much) less than 0.05. For example, the three highest adjusted p-values are as follows: 0.00051 (BRACID), 0.00035 (RIONA), 0.00020 (kNN). Anyway, one can conclude that RIONIDA is significantly better than any other algorithm (with a level of significance at 0.05). This is a quite astonishing result.

To sum up, for this strategy, the conclusions were very similar to those presented previously (for the *def* strategy). For more details, see [11].

5.1.3. Max *strategy for G-mean*

In Table 4, for each learning algorithm, the representative scores (for the *max* strategy) for all used data sets are given. One can see from Table 4 that for this strategy, still in most cases, the RIONIDA algorithm achieves the best score: for 20 data sets, 12 times it wins with all other algorithms. Again, the best outcome of the average rank for RIONIDA (1.725) is achieved. The difference between the average rank of the RIONIDA algorithm and the second-lowest average rank (3.90 for kNN) is still (compared to the result for the previous strategies) relatively high (2.175).

One can see from Table 4 that the Friedman statistical test gave again the p-value less than 10^{-7}. Moreover, in the discussed table, all the adjusted p-values are less than 0.05. It means that for each of the compared algorithms, even if it were possible to construct a meta-learning algorithm which for each data set would select the optimal algorithm configuration, it would be statistically worse than RIONIDA_G. It is worth underlining that this seems to be an impressive result.

For more details, see [11].

5.2. Comparison of RIONIDA with the selected state-of-the-art algorithms for F-measure

This subsection is analogous to Subsection 5.1 using F-measure instead of G-mean. Thus, RIONIDA is set to optimise the F-measure (such algorithm is called RIONIDA_F).

5.2.1. Def *strategy for F-measure*

In Table 5, for each learning algorithm, the representative scores of F-measure for all used data sets are given.

The RIONIDA algorithm was set to optimise F-measure, i.e. RIONIDA_F was used; hence, RIONIDA_F appears in the table instead of RIONIDA.

For half of the 20 data sets, RIONIDA wins with all other algorithms (has a rank equal to 1). RIONIDA again achieved the best average rank (2.15). It is smaller by 2.05 from the second-lowest average rank (4.2 for BRACID). The Friedman statistical test returns

Table 5
The values of F-measure (in %) for different algorithms and data sets for the *def* strategy. The RIONIDA algorithm was set to optimise F-measure (i.e. $\text{RIONIDA}_{\text{F}}$ was used). For each data set, the best-obtained score is shown in bold. Also, for illustration, for five algorithms on the right (including $\text{RIONIDA}_{\text{F}}$), ranks for these algorithms, and different data sets are shown (in parentheses). At the bottom are shown: (i) average rank for each algorithm, (ii) important outcomes of the Friedman statistical test (Friedman statistic, degrees of freedom, and p-value), and (iii) adjusted p-values (APV) with the Finner post-hoc test using RIONIDA as the control algorithm.

	The vectors of representative scores for F-measure chosen as perf. measure of our interest for the *def* str.									
Data set	kNN	PART	J48	RIPPER	RISE	MODLEM	MODLEM-C	RIONA	BRACID	$\text{RIONIDA}_{\text{F}}$
abalone	25.74	38.05	39.75	**42.35**	30.02	41.16 (2)	37.66 (5)	26.58 (9)	37.37 (6)	32.35 (7)
balance-scale	19.10	14.86	4.04	3.78	13.82	2.77 (9)	0.49 (10)	9.17 (6)	18.35 (3)	**34.30 (1)**
breast-cancer	44.78	39.80	40.78	39.41	44.31	42.90 (7)	44.79 (3)	43.05 (6)	45.52 (2)	**52.17 (1)**
breast-w	95.45	94.23	93.63	94.01	94.85	93.65 (8)	92.65 (10)	**96.39 (1)**	94.84 (5)	96.02 (2)
car	43.65	61.77	59.60	53.39	52.59	59.34 (6)	**85.88 (1)**	52.06 (9)	73.19 (3)	81.60 (2)
cleveland	33.33	34.55	36.04	37.75	31.55	35.35 (4)	16.80 (10)	35.01 (5)	33.36 (7)	**44.31 (1)**
credit-g	54.98	54.27	54.35	53.36	53.48	54.61 (5)	54.62 (3.5)	54.62 (3.5)	53.45 (9)	**58.27 (1)**
ecoli	59.63	57.66	56.93	60.05	59.52	59.28 (6)	53.05 (10)	58.40 (7)	59.87 (3)	**68.36 (1)**
glass	29.97	34.72	37.82	27.26	27.91	**38.28 (1)**	31.70 (5)	32.72 (4)	19.72 (10)	29.69 (7)
haberman	46.28	48.61	48.91	48.35	47.84	**50.06 (1)**	40.52 (10)	46.42 (7)	45.61 (9)	49.70 (2)
hepatitis	**60.64**	50.05	48.99	48.98	55.36	52.62 (6)	46.27 (10)	57.31 (4)	59.38 (3)	60.31 (2)
ionosphere	87.85	82.39	80.86	79.17	**88.71**	80.99 (8)	86.04 (6)	88.68 (2)	87.52 (4)	87.38 (5)
mammography	10.43	10.17	9.97	10.39	10.39	9.77 (9)	9.25 (10)	10.63 (3)	64.57 (2)	**67.33 (1)**
new-thyroid	94.82	90.77	91.49	90.94	92.44	89.85 (9)	88.12 (10)	95.36 (3)	**96.91 (1)**	96.40 (2)
nursery	53.08	91.54	74.03	68.46	75.36	95.40 (3)	**99.73 (1)**	75.74 (6)	95.10 (4)	98.92 (2)
pima	63.03	63.43	63.28	64.26	63.69	63.01 (8)	62.24 (10)	62.39 (9)	65.82 (2)	**66.04 (1)**
postoperative	24.13	19.73	20.99	17.49	20.10	18.59 (8)	23.55 (4)	17.65 (9)	31.93 (2)	**33.61 (1)**
transfusion	45.78	45.39	45.92	46.84	46.15	45.33 (9)	39.12 (10)	46.06 (5)	47.08 (2)	**50.02 (1)**
vehicle	84.44	86.08	85.14	85.96	83.55	84.44 (8.5)	89.74 (2)	86.10 (3)	85.81 (6)	**89.79 (1)**
yeast	37.77	33.98	35.40	37.37	40.03	37.32 (7)	29.00 (10)	38.14 (4)	**41.62 (1)**	41.24 (2)
average rank	5.475	6.1	6.3	6.425	5.825	6.225	7.025	5.275	4.2	2.15
Friedman test	Friedman's chi-squared = 38.785, df = 9, p-value = $1.26 \cdot 10^{-5}$									
APV Finner	0.00066	0.00007	0.00004	0.00004	0.00019	0.00005	$< 10^{-5}$	0.00124	0.03226	control

again (much) less p-value than 0.05. Also, all the adjusted p-values of the Finner procedure are less than 0.05. Thus, we can claim that RIONIDA is significantly better than any other algorithm used in the comparison. However, the highest p-value (around 0.03 for RIONIDA) is close to the threshold of 0.05. It shows that RIONIDA outperforms BRACID not as evidently as in the case for G-mean. Probably this is because BRACID was implemented to optimise F-measure. The second-highest p-value is for RIONA (with its optimal parameter settings and filter) and is smaller than 10^{-2}. All other p-values (related to other algorithms) are smaller than 10^{-3}. More details can be found in [11].

5.2.2. Opt *strategy for F-measure*

For this strategy, the conclusions are very similar to those presented previously (for the *def* strategy). RIONIDA is significantly better than any other compared algorithm. However, RIONIDA achieves a little bit worse scores (in relation to other algorithms) and ranks than for the *def* strategy for F-measure. For details, see [11].

5.2.3. Max *strategy for F-measure*

In Table 6, for each learning algorithm, the representative scores (for the *max* strategy) for all used data sets are given. One can see from this table that for this strategy, the RIONIDA algorithm wins with other algorithms 6 times. RIONIDA again achieved the best average rank (2.85). It is smaller by 1.6 from the second-lowest average rank (4.45 for kNN). The Friedman statistical test again shows the statistical differences among compared algorithms (with a p-value much less than 0.05).

All the adjusted p-values of the Finner procedure but one are less than 0.05. The only exception was kNN: RIONIDA achieved better average rank (2.85) than kNN (4.45) but the difference between RIONIDA and kNN is not statistically significant for scores computed in such a way (adjusted p-value in the Finner statistical test for kNN was 0.09). However, one should also bear in mind that real meta-learning algorithm using kNN would obtain worse results than we took into comparison (as it was mentioned, we took maximal values of possible scores).

To sum up, for the *max* strategy, RIONIDA achieves noticeably worse scores (in relation to other algorithms) and ranks than for the *opt* strategy. However, the statistical conclusion concerning comparisons with other algorithms remains nearly unchanged: RIONIDA is significantly better than each of the compared algorithms, excluding kNN.

For more details, see [11].

5.3. Conclusions for experiments

To sum up, we performed experiments to compare our new proposed RIONIDA algorithm with the selected nine state-of-the-art algorithms with the possible use of state-of-the-art configurations of filters. Both for G-mean and F-measure and for all three considered

Table 6
The values of F-measure (in %) for different algorithms and data sets for the *max* strategy (additionally for RIONIDA ranks are shown in parentheses). The RIONIDA algorithm was set to optimise F-measure (i.e. $\text{RIONIDA}_{\text{F}}$ was used). It should be noted that for both $\text{RIONIDA}_{\text{F}}$ and BRACID, one default algorithm configuration was used (i.e. no filter and default parameters of the algorithm were used) – their scores are the same as in the *def* strategy). For the other learning algorithms, the vector of representative scores was generated using the *max* strategy. For each data set, the best-obtained score is shown in bold. At the bottom are shown: (i) average rank for each algorithm, (ii) important outcomes of the Friedman statistical test (Friedman statistic, degrees of freedom, and p-value), and (iii) adjusted p-values (APV) with the Finner post-hoc test using RIONIDA as the control algorithm.

	The vectors of representative scores for F-measure chosen as perf. measure of our interest for the *max* str.									
Data set	kNN	PART	J48	RIPPER	RISE	MODLEM	MODLEM-C	RIONA	BRACID	$\text{RIONIDA}_{\text{F}}$
abalone	26.85	38.88	40.17	**43.40**	32.58	41.16	37.86	26.58	37.37	32.35 (8)
balance-scale	23.91	19.81	4.22	4.40	13.82	2.77	0.50	9.17	18.35	**34.30 (1)**
breast-cancer	47.95	42.10	42.00	45.32	44.81	45.24	44.79	47.49	45.52	**52.17 (1)**
breast-w	95.75	94.32	93.69	94.18	95.39	93.80	92.65	**96.39**	94.84	96.02 (2)
car	50.49	**91.76**	76.43	67.12	68.37	88.34	88.34	77.18	73.19	81.60 (4)
cleveland	40.21	38.57	36.10	38.94	31.55	35.35	16.80	35.01	33.36	**44.31 (1)**
credit-g	54.98	54.72	54.35	53.68	53.48	54.61	54.64	54.62	53.45	**58.27 (1)**
ecoli	**69.20**	60.65	62.76	64.70	61.54	61.64	53.13	62.20	59.87	68.36 (2)
glass	32.87	40.55	42.86	33.62	29.86	**47.21**	31.89	35.68	19.72	29.69 (9)
haberman	46.72	48.61	49.00	48.81	47.84	**50.06**	40.52	46.42	45.61	49.70 (2)
hepatitis	**64.27**	53.86	52.80	49.99	55.36	53.27	46.90	57.31	59.38	60.31 (2)
ionosphere	90.65	86.76	85.32	85.81	**91.12**	87.08	87.08	89.19	87.52	87.38 (5)
mammography	65.52	60.84	63.30	65.02	**67.83**	67.30	62.26	60.84	64.57	67.33 (2)
new-thyroid	95.33	92.92	91.61	91.87	95.56	92.30	89.57	95.85	**96.91**	96.40 (2)
nursery	57.43	99.62	87.59	73.34	95.28	**99.73**	**99.73**	98.98	95.10	98.92 (5)
pima	63.43	63.48	63.34	64.26	63.69	63.34	62.74	62.39	65.82	**66.04 (1)**
postoperative	**38.19**	21.70	21.67	18.02	20.10	18.59	23.55	17.65	31.93	33.61 (2)
transfusion	46.10	48.95	47.67	46.84	46.15	45.33	40.18	46.06	47.08	**50.02 (1)**
vehicle	88.15	89.77	87.55	88.98	86.55	89.70	**90.22**	90.16	85.81	89.79 (3)
yeast	41.60	34.77	37.24	39.98	40.29	37.32	29.00	39.57	**41.62**	41.24 (3)
average rank	4.45	5.175	6.575	6	5.85	5.45	7.325	5.675	5.65	**2.85**
Friedman test	Friedman's chi-squared = 28.68, df = 9, p-value = 0.0007337									
APV Finner	0.09469	0.01705	0.00045	0.00300	0.00388	0.00850	0.00003	0.00570	0.00570	control

strategies, it was shown that RIONIDA significantly outperforms any algorithm used in the comparison for selected imbalanced data sets (with one exception for the *max* strategy). The obtained experimental results seem to be exceptionally good.

In particular, RIONIDA outperforms RIONA regardless of the used filter and other settings for RIONA. This confirms that the performance of RIONIDA cannot be obtained by using RIONA with proper settings and filters. RIONIDA is a more compound algorithm, and its results cannot be obtained by simple modifications of RIONA use.

6. Discussion

In this section, we present some additional comments on the performed experiments, which can help to understand why the RIONIDA algorithm outperforms some well-known methods dealing with imbalanced data. At the same time, we explain some advantages of RIONIDA. Finally, we analyse the real running time of RIONIDA, which was measured during the experiments presented in the previous section. In particular, we present a comparison of it with other algorithms used in the experiments.

6.1. *Studying the role of RIONIDA components*

The key component of RIONIDA is the estimation of its performance for each possible triple of internal parameters $(k, p, s) \in K \times P \times S$ (see Section 4.4). The significance of the parameters k, p, s was shown in Subsections 4.3.2, 4.3.3, 4.3.4, respectively. The estimation of the optimal values of these parameters is done very precisely in the learning phase since the validation process is performed by the leave-one-out method on the whole training set. Thus, in a sense, the full information for the given training set is used in the process of tuning these internal parameters. It is worth mentioning that the time complexity of this process is relatively low due to using the dynamic programming technique. All this means that the RIONIDA algorithm can learn the relevant values of its internal parameters very efficiently and precisely (and so proves to be highly effective). In our opinion, this is one of the main advantages of the RIONIDA algorithm.

6.2. *The* balance-scale *data set and outliers*

Out of the data sets used in the experiments, we would like to turn out the readers' attention to the *balance-scale* data set. For most of the objects from the minority class of this data set, the objects closest to them belong to the majority class. Such objects are called outliers. In other words, in the considered data set, most of the objects from the minority class are outliers. This is the reason why this data set is considered as a very hard imbalanced learning problem in [31].

Also, we performed separate experiments for RIONIDA with the fixed parameter $s = 1.0$ (which relates to the pure rule-based approach). In this case, for the minority class, generally, no (consistent) rules were found. The fact that most of the objects are outliers explains this fact. It also provides an intuition as to why algorithms that use standard rules may construct classifiers of poor quality for such data sets.

However, RIONIDA in most of the cross-validation splits (in a quite stable way – see next subsubsection) finds the optimal value $s = 0.5$. This corresponds to the situation that original rules may be inconsistent, but after changing the coverage region of the rule by half are becoming consistent. Generally, if we take into account objects from the minority class, the rules with decreasing value of the parameter s enable us to increase the Sensitivity of the rule.

It is an example illustrating that the RIONIDA algorithm can deal with data sets containing many outliers. This fact can be regarded as a powerful advantage of the RIONIDA algorithm.

6.3. Analysis of the optimal values of parameters obtained in the learning phase of RIONIDA

During the experiments presented in the previous section, we saved the internally learned optimal values of the parameters k, p, and s obtained during the learning phase in different runs of the performed experiments (10 times repeated 10-fold stratified cross-validation process) for RIONIDA. Thus, we obtained 100 triples of the optimal parameter values.

It can be informative to check for particular data sets whether the learned optimal parameters are 'stable' in different runs of RIONIDA. This can be relevant for at least three reasons.

First, stability (of one or more parameters) for a fixed domain may indicate that specific values of parameters are appropriate globally for all objects from that domain. This may mean that for future additional training (currently unknown) objects from that domain, no further learning of (one or more) parameters is needed. Also, small fluctuations of (one or more) optimal values of parameters may indicate that for future training objects (currently unknown) from that domain, the learning could be limited to a smaller range of some (one or more) parameters. Such limitations can influence the learning time and space allocations of the algorithm. This can be essential for the scalability of the RIONIDA algorithm. For example, this can be crucial for the application of RIONIDA to so-called big data (see e.g. [42]). In the considered case of stability of parameters, learning of the optimal parameters could be done for a relatively small part of the data set.

Second, the stability of (one or more) parameters can be an argument for the quality of the obtained classifier. The more stable the optimal value of the parameter is, the more reliable the resulting classifier can be regarded as.

Third, in the case of stability (of one or more parameters), the values of stable parameters may be a kind of description of a domain. For example, a domain can be described as 'more appropriate for rule-based methods' or 'more appropriate for instance-based methods'.

In Table 7, we present the averages and standard deviations of the optimal values of the parameters k, p, and s obtained in the mentioned experiments for RIONIDA (for both $\text{RIONIDA}_{\text{G}}$ and $\text{RIONIDA}_{\text{F}}$). In the current analysis, we focus on $\text{RIONIDA}_{\text{G}}$, and therefore RIONIDA will refer to this setting. In current considerations, the most interesting for us is the standard deviation. In this case, the small value of this measure means that in most (or all) runs of RIONIDA, the learned optimal values of the parameter were similar (or even equal).

Let us describe some conclusions for a few exemplary data sets and information from this table (and also from direct observations of the learned optimal values of parameters).

For *balance-scale* data set, the learned optimal parameters seem 'the most stable'. In all runs of RIONIDA, the learned optimal values of the parameters k and s were equal to 9 and 0.5, respectively. The learned optimal value of the parameter p was around value 0.1. It was equal to 0.08 (15 times), 0.09 (31), 0.1 (8), 0.11 (11), 0.12 (34), or 0.13 (1 time).

For *hepatitis*, *ionosphere*, *transfusion*, and *vehicle* data sets, the learned optimal parameter s in all considered runs of RIONIDA was constant and equal to -0.1. As $s = -0.1$ corresponds to the pure instance-based method in RIONIDA (see Subsection 4.3.4), one can describe these data sets as 'more appropriate for instance-based methods'. For *new-thyroid* data set, the value of the optimal parameter s can be considered as very stable (98 times it was equal to -0.1, once to 0.5, and once to 0.9). Also, for *breast-w* data set, the value of the optimal parameter s can be considered as very stable (99 times it was equal to -0.1, and once to 1.0). Hence, these two data sets can also be described as 'more appropriate for instance-based methods'.

On the other hand, *yeast* data set has a very stable value of the optimal parameter s around value 1.0. It was equal to 1.0 in 98 cases, 0.5 in one case, and 0.7 in one case. As $s = 1.0$ corresponds to the pure rule-based method in RIONIDA (see Subsection 4.3.4), this data set can be described as 'more appropriate for rule-based methods'. For *abalone* data set the value of the optimal parameter s can be considered as very stable around value 0.9. It was equal to 1 (81 times), 0.9 (5), 0.8 (3), 0.7 (7), 0.6 (1), 0.5 (1), 0.4 (1), or 0.3 (1 time). Also, this data set can be described as 'more appropriate for rule-based methods'.

As it was mentioned, for *balance-scale* data set, the value of the optimal parameter s was constant for all considered runs of RIONIDA and was equal to 0.5. It is an interesting example for the data set which can be described as 'data set appropriate for methods between instance- and rule-based methods'. Another example of such a case is *haberman* data set. Its optimal value of the parameter s was around value 0.6. It was equal to 0.6 (28 times), 0.5 (20), 0.7 (2), 0.4 (5), 0.8 (4), 0.3 (9), 0.9 (28), and 0.2 (4 times). See also Subsection 4.3.4 for considerations on *haberman* data set (and Fig. 4) taking into account the whole available data and dependence of G-mean on the parameter s.

Now, let us take into account the fluctuations of the optimal values of the parameter p. As can be seen in Table 7, for most of the considered data sets, this value has a relatively small standard deviation (for *yeast*, *abalone*, *mammography*, *balance-scale*, *credit-g*,

Table 7
Table presenting fluctuations of optimal values of parameters k, p, s among different runs (different splits in the cross-validation schemes) of RIONIDA (RIONIDA$_G$ and RIONIDA$_F$) for each data set used in experiments. Averages and standard deviations of optimal parameters k, p, and s are rounded to integers, two decimals, and one decimal, respectively.

	RIONIDA$_G$			RIONIDA$_F$		
	Averages and standard deviations of optimal parameters					
Data set	k	p	s	k	p	s
abalone	76 ± 23	0.08 ± 0.01	0.9 ± 0.1	53 ± 21	0.15 ± 0.02	1.0 ± 0.1
balance-s.	9 ± 0	0.10 ± 0.02	0.5 ± 0.0	9 ± 1	0.13 ± 0.02	0.5 ± 0.0
breast-c.	67 ± 22	0.28 ± 0.03	0.2 ± 0.3	67 ± 21	0.28 ± 0.03	0.2 ± 0.3
breast-w	16 ± 22	0.12 ± 0.09	-0.1 ± 0.1	17 ± 23	0.13 ± 0.09	-0.1 ± 0.1
car	33 ± 29	0.18 ± 0.14	0.4 ± 0.4	59 ± 37	0.24 ± 0.12	0.8 ± 0.4
cleveland	62 ± 22	0.12 ± 0.02	0.4 ± 0.6	81 ± 14	0.16 ± 0.02	0.8 ± 0.4
credit-g	62 ± 19	0.30 ± 0.02	0.6 ± 0.4	60 ± 19	0.30 ± 0.02	0.6 ± 0.4
ecoli	81 ± 28	0.26 ± 0.03	0.7 ± 0.3	27 ± 11	0.37 ± 0.03	0.7 ± 0.4
glass	12 ± 8	0.08 ± 0.05	0.2 ± 0.4	4 ± 5	0.07 ± 0.11	0.0 ± 0.2
haberman	81 ± 18	0.19 ± 0.03	0.6 ± 0.2	80 ± 16	0.18 ± 0.03	0.6 ± 0.2
hepatitis	35 ± 14	0.14 ± 0.03	-0.1 ± 0.0	31 ± 17	0.18 ± 0.06	-0.1 ± 0.1
ionosphere	7 ± 3	0.02 ± 0.03	-0.1 ± 0.0	7 ± 3	0.06 ± 0.06	-0.1 ± 0.0
mammogr.	73 ± 31	0.03 ± 0.01	0.8 ± 0.3	8 ± 3	0.26 ± 0.04	0.1 ± 0.4
new-t.	67 ± 22	0.11 ± 0.03	-0.1 ± 0.1	66 ± 22	0.11 ± 0.03	-0.1 ± 0.1
nursery	33 ± 33	0.23 ± 0.13	0.8 ± 0.5	27 ± 35	0.17 ± 0.17	0.5 ± 0.6
pima	44 ± 21	0.32 ± 0.03	0.8 ± 0.4	60 ± 20	0.29 ± 0.03	0.8 ± 0.4
postop.	13 ± 12	0.19 ± 0.09	0.3 ± 0.5	17 ± 12	0.16 ± 0.08	0.0 ± 0.3
transfusion	28 ± 15	0.24 ± 0.02	-0.1 ± 0.0	31 ± 14	0.28 ± 0.04	-0.1 ± 0.0
vehicle	6 ± 3	0.21 ± 0.13	-0.1 ± 0.0	4 ± 3	0.28 ± 0.19	-0.1 ± 0.0
yeast	54 ± 19	0.03 ± 0.01	1.0 ± 0.1	37 ± 21	0.21 ± 0.03	0.6 ± 0.5

cleveland, *transfusion*, *breast-cancer*, *hepatitis*, *haberman*, *new-thyroid*, *ionosphere*, *pima*, and *ecoli* data sets). In these cases, the learned optimal value of the parameter p can be considered as stable.

What about the cases when the optimal values of parameters are unstable? These could be investigated in two directions.

First, one could check how fluctuations of the optimal values of parameter change the value of considered performance measure. This issue was somehow investigated globally (for the whole data sets) in Subsections 4.3.2, 4.3.3, 4.3.4. However, further investigation could be done.

Second, such fluctuations may suggest that searching not globally but locally for the optimal values of parameters (separately for different regions of a considered domain) could potentially increase the quality of RIONIDA performance.

Let us now concentrate on the average value of the optimal parameter p from Table 7. It should be noted that for many considered data sets (11 out of 20) this value is very close to the percentage of the minority class in the data set (see Table 2). (Moreover, for these 11 data sets except *breast-cancer*, these differences are less than the standard deviation of the value of the optimal parameter p.) The distance between these two values is:

- less than 1% for *glass*, *yeast*, *credit-g*, *mammography*, *transfusion*, *abalone*, and *cleveland* data sets;
- less than 3% for *breast-cancer*, *balance-scale*, *pima*, and *vehicle* data sets.

The presented observations are consistent with the theorem formulated and proved in [11] (see also [26]). This fact could be used in future research for using the default candidate for the optimal value of the parameter p. This fact could also be used to search for the optimal value of this parameter around the default value. Such an approach could be especially useful for big data sets.

On the other hand, it should be noted that there exist a few data sets for which the difference between the average optimal value of the parameter p and the percentage of the minority class in data set is relatively high: *ionosphere* (approximately 34% of difference), *breast-w* (22%), *nursery* (20%), *ecoli* (15%). This fact proves the usefulness of learning the optimal value of the parameter p in general (see also Subsection 4.3.3).

6.4. Analysis of running time of RIONIDA

In Subsection 4.5.2 we calculated the time complexity of the learning phase of RIONIDA. Here, we check whether the time of the learning phase in practice reaches the (pessimistic) theoretical time complexity. In the performed experiments, the number of conditional attributes was relatively small (from 4 to 34). Also, the sets P and S (i.e. sets of admissible values of the parameters p, s) are constant in our comparative experiment. Thus we omit these three factors in the analysis of the time of learning phase. In the performed experiments, the number of objects in data sets varied from 90 to 12960 examples in total. We make the running time analysis only for this factor.

Fig. 5 presents the relationship between the size of the data set and the learning phase duration of RIONIDA. Each data set is represented in this figure by point (x, y), where x = number of examples in data set, y = time of learning phase.

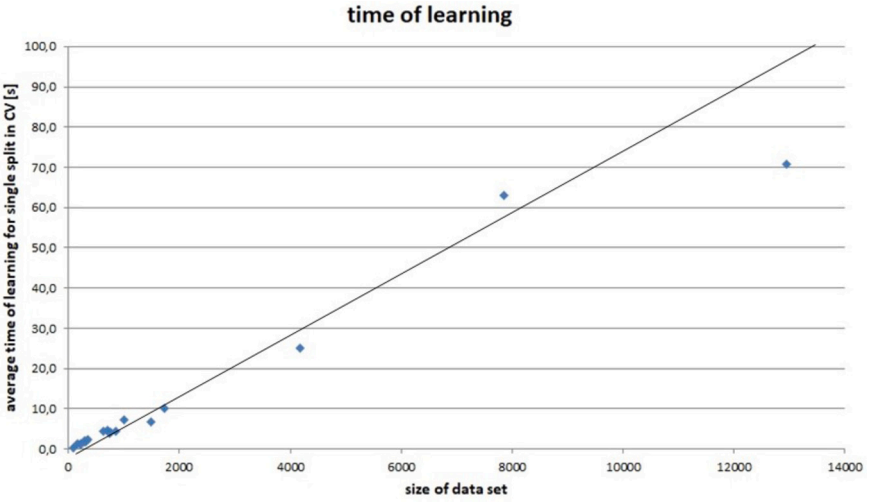

Fig. 5. Dependence of the learning phase duration of RIONIDA (in seconds) on the size of data set (i.e. the number of objects in data set). The figure presents the average time of the learning phase of RIONIDA for a single split in the 10-fold stratified cross-validation. In any split, the training set contains roughly 90% of the data set.

It is visible, that the points roughly lay on a straight line. At first glance, it is a surprising observation since the (pessimistic) theoretical time complexity is a quadratic function of the number of training examples. Below we explain it and add further comments.

Let us recall that the time complexity of the learning phase is $O(mn^2 + n|S| \cdot k_{max} \cdot (mk_{max} + |P|))$, where $n = |trnSet|$, $m = |A|$, k_{max} is the parameter used to define the maximal size of the neighbourhood to be analysed ($k_{max} = |K|$), P, S are sets of admissible values of the parameters p, s, respectively (see Theorem 1 in Subsection 4.5.2). In our primary experiments $|P| \leq mk_{max}$ holds, and as a consequence, the time complexity is $O(m(n^2 + n|S| \cdot k_{max}^2))$. For the data sets used in our experiments $n < 13000$. Since in our primary experiments $|S| = 12$, $k_{max} = 100$, then $n^2 < n|S| \cdot k_{max}^2$. In other words, for the used data sets and settings, the factor $n|S| \cdot k_{max}^2$ is dominant over the factor n^2. This fact explains the observed in the performed experiments the 'linearity' of the time of learning phase relative to n. The quadratic factor will become dominant for $n > 120000$.

On the other hand, the quadratic time complexity relates to the searching for k_{max} nearest objects to the considered training example among n objects (see Subsection 4.5.2). We assumed that this operation could be done in the linear time relative to n (see Subsection 4.5.2). However, in our implementation, we use indexing trees to speed up this operation [43]. It was experimentally shown in [43, pp. 77-78] that by using indexing trees: (1) this operation is faster than linear, (2) the acceleration (of this operation) significantly grows with growing n. Also, it was (experimentally) shown that the time of constructing indexing trees is significantly shorter than the time of (multiple) searching of nearest neighbours in a data set. All these facts were not analysed theoretically; thus, we can only say that in our implementation, instead of factor n^2, occurs a factor with time complexity between linear and quadratic. This fact appears promising in the context of a potential need for scalability of RIONIDA.

We also analysed the testing time of a single object for each data set. The average time of testing of a single object for different data sets was between 0.03 ms and 0.35 ms (parts of milliseconds).

We observed that the average time of testing of a single object for the larger data sets used in our experiments (*abalone* and *nursery*) is comparable to the case for the smaller data sets. Thus, we repeated the experiments without using indexing trees to check whether significant acceleration is achieved by using this specialised data structure.

Fig. 6 shows the average time of testing of a single object for (1) standard version of RIONIDA with use of indexing trees, and (2) version of RIONIDA without using indexing trees. In the case without the use of indexing trees, one can observe an approximately linear dependence of the average time for checking a single object on the size of the data sets. This observation is consistent with the theoretical time complexity of testing operation for RIONIDA (see Subsection 4.5.1). On the other hand, for the version with the use of indexing trees, one can observe variability between two constant values. The plot in this figure suggests dependence close to a constant value. However, we must admit that we used rather small data sets to draw any far-reaching conclusions about the (experimental) dependence of the average time of testing of a single object (for RIONIDA) on the size of data sets. Regardless, this shows that even for data sets used in our experiments, which are relatively small, the significant acceleration of the testing phase for RIONIDA by using indexing trees is achieved. This is a promising fact for attempting to analyse big data sets.

Taking into account the above considerations, let us come back shortly to the case of the time of learning phase. It should be noted that for the testing phase, the operation of searching for the nearest objects is dominant. As it was mentioned, this (repeated) operation will become dominant for larger data sets also in the learning phase. If this operation for larger data sets would also take time close to constant, it would strongly affect the time complexity of the learning phase. Thus, the observations for the considered data sets justify the mentioned supposition that, for the learning phase, the practical time complexity can be close to linear. To be more precise, this issue needs further investigation in the future.

Finally, we compare the time of computations for different learning algorithms. For details of used algorithms and filter settings in this comparison, see [11].

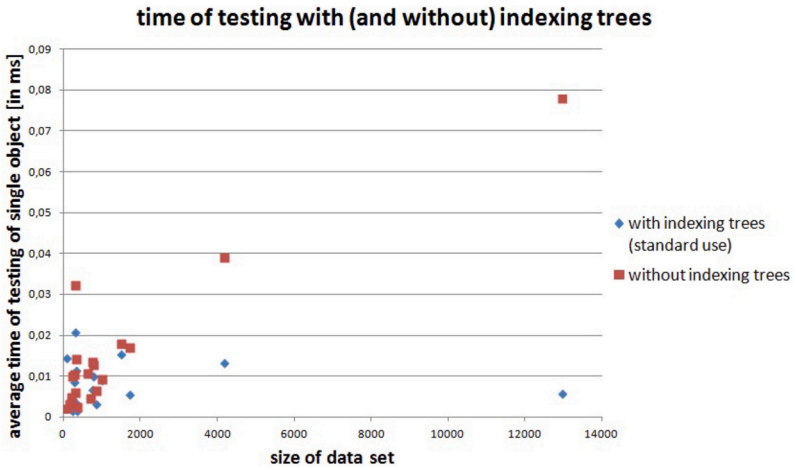

Fig. 6. Dependence of the average time of testing of a single object (in milliseconds) on the size of data set (number of objects in data set) for: (1) standard version of RIONIDA with use of indexing trees, and (2) version of RIONIDA without using indexing trees.

By means of training time (generally the most time-consuming phase), RIONIDA is comparable to other algorithms (25 s[10] for MODLEM-C, 426 s for RISE, and between 177 s and 238 s for others) used in experiments (see [11] for the appropriate figure). However, precisely, RIONIDA (156 s) is on the second place, after MODLEM-C (which performs a few times faster). Moreover, it is worth recalling that the learning phase of the used in experiments implementation of RIONIDA can be accelerated a few times (see [11]).

RIONIDA (252 ms[11]) is in the seventh place by means of testing time. It is around 50 times slower than PART (4 ms), J48 (5 ms), and RIPPER (5 ms) algorithms; around 4 times slower than MODLEM (69 ms), MODLEM-C (52 ms), and RIONA (54 ms). On the other hand, it is more than 150 times faster than kNN (40495 ms), and a few times faster than RISE (695 ms) and BRACID (1256 ms) (see [11] for the appropriate figure).

Taking into account that usually training phase is dominant, the RIONIDA algorithm, on average, has a comparable time of computations to the other learning algorithms (possibly combined with relevant filters) used in the experiments.

7. Conclusions

RIONIDA is an extension of RIONA combining the instance- and rule-based approaches for imbalanced data. Additionally, RIONIDA uses rules that are more general than the ones used in RIONA. This algorithm optimises the explicitly given performance measure. All main ideas embedded in RIONIDA are essential for obtaining a high performance quality, including optimising the fixed performance measure as well as three proposed internal parameters. This algorithm is relatively fast (both in the training and testing phase). Furthermore, the theoretical results concerning the parameter responsible for assigning relevant weights for the minority and majority classes can be used to speed up the training phase. The RIONIDA algorithm (as RIONA) has the desired property of explainability. RIONIDA achieves impressively good results compared to the quality of the other state-of-the-art algorithms analysed in the article. RIONIDA significantly outperforms these algorithms in terms of the selected performance measures. Furthermore, the running time of RIONIDA is comparable to that of the algorithms used. It was far more successful to construct the RIONIDA algorithm than to use the RIONA algorithm with filters for imbalanced data or different settings.

In summary, the main advantages of RIONIDA are as follows. In particular, (i) it is suitable for imbalanced data (also for "balanced data"), (ii) its decisions are explainable to the user, (iii) the reasoning behind the algorithm construction is explainable to the user, (iv) it shows good performance on complex imbalanced data by optimising the quality measure, class weight, scalability parameter, and the size of the neighbourhoods, (v) the user can specify the optimised quality measure based on the confusion matrix, (vi) the learning phase is efficient due to the application of dynamic programming. The drawbacks of RIONIDA are as follows: (i) it is not a multiclass-classifier (for two decisions only), (ii) it learns only simple features, and (iii) the explanation of good experimental performance on complex imbalanced data is not theoretically justified.

There are several possible directions for future research related to: (1) extensions of RIONIDA, (2) continuation of the presented experiments, and (3) application of RIONIDA to more complex tasks [11].

In particular, preliminary works suggest that there is potential that the use of RIONIDA for imbalanced big data can also be achievable and successful. However, extensive trials should be carried out to determine the practical use of RIONIDA in this case.

[10] This and other presented run times are the summed (for all data sets used in experiments) average times of the training phase (sum of tho phases: filtering when used and training by learning algorithm) for a single split in 10-fold stratified cross-validation.

[11] This and other presented run times are the summed (for all data sets used in experiments) average times of the testing phase for a single split in 10-fold stratified cross-validation.

Required would also be the solution for the case of imbalanced data sets with multiple classes. One may solve such a classification problem by transforming it into a family of binary classification subproblems and appropriately joining the partial solutions (see e.g. [44,45]). In addition, we believe that our solution could potentially be extended by using different weights for each decision class. It requires further rethinking of our ideas and more research in these areas.

In RIONIDA, new features are used by grouping symbolic and numerical attributes. In the future, we also plan to extend RIONIDA with techniques for hierarchical learning of new features. In particular, we want to extend RIONIDA with techniques that search for new features using a rough-set-based deep learning approach. Specifically, incorporating granulation at different layers may help reduce the size of the network and decrease learning time [46].

In further research, we plan to compare RIONIDA with deep learning methods developed for the analysis of imbalanced data. In particular, we plan to extend the comparison of RIONIDA with methods using data augmentation based on deep neural networks [47]. Moreover, we would like to explore the possibility of combining RIONIDA with deep learning methods.

RIONIDA shows good performance on complex imbalanced data. In our further study, we plan to look for an explanation of why this happens, using the recent work dedicated to analysing the joint effect of class imbalance and overlap [48–50]. In addition, we plan to further investigate the problem of explainability by RIONIDA using recent explainability results related to deep learning methods for imbalanced data analysis [22].

CRediT authorship contribution statement

Grzegorz Góra: Writing – original draft. **Andrzej Skowron:** Writing – original draft.

Declaration of competing interest

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

Data availability

Data will be made available on request.

References

[1] W. Chen, K. Yang, Z. Yu, Y. Shi, C.L.P. Chen, A survey on imbalanced learning: latest research, applications and future directions, Artif. Intell. Rev. 57 (6) (2024) 137, https://doi.org/10.1007/S10462-024-10759-6.
[2] G. Aguiar, B. Krawczyk, A. Cano, A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework, Mach. Learn. 113 (7) (2024) 4165–4243, https://doi.org/10.1007/S10994-023-06353-6.
[3] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284, https://doi.org/10.1109/TKDE.2008.239.
[4] V. López, A. Fernández, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics, Inf. Sci. 250 (2013) 113–141, https://doi.org/10.1016/j.ins.2013.07.007.
[5] K. Napierała, Improving Rule Classifiers for Imbalanced Data, Ph.D. thesis, Poznań University of Technology, Poznań, 2012.
[6] N. Japkowicz, S. Stephen, The class imbalance problem: a systematic study, Intell. Data Anal. 6 (5) (2002) 429–449, https://doi.org/10.3233/IDA-2002-6504.
[7] G.M. Weiss, F. Provost, Learning when training data are costly: the effect of class distribution on tree induction, J. Artif. Intell. Res. 19 (1) (2003) 315–354, https://doi.org/10.1613/jair.1199.
[8] G.M. Weiss, The impact of small disjuncts on classifier learning, in: R. Stahlbock, S.F. Crone, S. Lessmann (Eds.), Data Mining: Special Issue in Annals of Information Systems, Springer, Boston, MA, 2010, pp. 193–226, https://doi.org/10.1007/978-1-4419-1280-0_9.
[9] H. He, Y. Ma (Eds.), Imbalanced Learning: Foundations, Algorithms, and Applications, 1st edition, Wiley-IEEE Press, Piscataway, NJ, 2013, https://doi.org/10.1002/9781118646106.
[10] G. Góra, A. Wojna, RIONA: a new classification system combining rule induction and instance-based learning, Fundam. Inform. 51 (4) (2002) 369–390.
[11] G. Góra, Combining instance-based learning and rule-based methods for imbalanced data, Ph.D. thesis, University of Warsaw, Warsaw, 2022, https://www.mimuw.edu.pl/sites/default/files/gora_grzegorz_rozprawa_doktorska.pdf.
[12] Weka 3: Machine Learning Software in Java, https://www.cms.waikato.ac.nz/ml/weka/.
[13] Rseslib 3: rough set and machine learning open source in Java, http://rseslib.mimuw.edu.pl.
[14] G. Góra, A. Skowron, RIONIDA: a novel algorithm for imbalanced data combining instance-based learning and rule induction, in: M. Hu, C. Cornelis, Y. Zhang, P. Lingras, D. Ślęzak, J. Yao (Eds.), Rough Sets - International Joint Conference, IJCRS 2024, Halifax, Canada, May 17-20, 2024, Proceedings, Part I, in: Lecture Notes in Computer Science, vol. 14839, Springer, Cham, 2024, pp. 201–219, https://doi.org/10.1007/978-3-031-65665-1_13.
[15] P. Domingos, Unifying instance-based and rule-based induction, Mach. Learn. 24 (2) (1996) 141–168, https://doi.org/10.1007/BF00058656.
[16] J. Li, G. Dong, K. Ramamohanarao, L. Wong, DeEPs: a new instance-based lazy discovery and classification system, Mach. Learn. 54 (2) (2004) 99–124, https://doi.org/10.1023/B:MACH.0000011804.08528.7d.
[17] S. Salzberg, A nearest hyperrectangle learning method, Mach. Learn. 6 (1991) 251–276, https://doi.org/10.1007/BF00114779.
[18] D. Wettschereck, T.G. Dietterich, An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms, Mach. Learn. 19 (1995) 5–27, https://doi.org/10.1007/BF00994658.
[19] D.W. Aha (Ed.), Lazy Learning, 1st edition, Springer, Dordrecht, 1997, https://doi.org/10.1007/978-94-017-2053-3.
[20] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 2nd edition, The MIT Press and McGraw-Hill Book Company, Cambridge, MA, 2001.
[21] Z. Pawlak, Rough sets, Int. J. Comput. Inf. Sci. 11 (1982) 341–356, https://doi.org/10.1007/BF01001956.
[22] D. Dablain, C. Bellinger, B. Krawczyk, D.W. Aha, N.V. Chawla, Understanding imbalanced data: XAI & interpretable ML framework, Mach. Learn. 113 (6) (2024) 3751–3769, https://doi.org/10.1007/S10994-023-06414-W.
[23] V.N. Vapnik, Statistical Learning Theory, 1st edition, Wiley-Interscience, New York, NY, 1998.
[24] M. Bekkar, H.K. Djemaa, T.A. Alitouche, Evaluation measures for models assessment over imbalanced data sets, J. Inf. Eng. Appl. 3 (10) (2013) 27–38.

[25] N. Japkowicz, M. Shah, Evaluating Learning Algorithms: A Classification Perspective, Cambridge University Press, Cambridge, 2011, https://doi.org/10.1017/CBO9780511921803.
[26] G. Góra, A. Skowron, On kNN class weights for optimising G-mean and F1-score, in: A. Campagner, O. Urs Lenz, S. Xia, D. Ślęzak, J. Wąs, J. Yao (Eds.), Rough Sets, in: Lecture Notes in Computer Science, vol. 14481, Springer, Cham, 2023, pp. 414–430, https://doi.org/10.1007/978-3-031-50959-9_29.
[27] D.J. Hand, Measuring classifier performance: a coherent alternative to the area under the ROC curve, Mach. Learn. 77 (1) (2009) 103–123, https://doi.org/10.1007/s10994-009-5119-5.
[28] T. Raeder, G. Forman, N.V. Chawla, Learning from imbalanced data: evaluation matters, in: D.E. Holmes, L.C. Jain (Eds.), Data Mining: Foundations and Intelligent Paradigms: Volume 1: Clustering, Association and Classification, Springer, Heidelberg, 2012, pp. 315–331.
[29] M. Lichman, UCI machine learning repository, http://archive.ics.uci.edu/ml, 2013.
[30] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357, https://doi.org/10.1613/jair.953.
[31] K. Napierała, J. Stefanowski, Types of minority class examples and their influence on learning classifiers from imbalanced data, J. Intell. Inf. Syst. 46 (3) (2016) 563–597, https://doi.org/10.1007/s10844-015-0368-1.
[32] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Stat. Assoc. 32 (200) (1937) 675–701, https://doi.org/10.1080/01621459.1937.10503522, arXiv:https://www.tandfonline.com/doi/pdf/10.1080/01621459.1937.10503522.
[33] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30, https://doi.org/10.5555/1248547.1248548.
[34] J.W. Grzymala-Busse, L.K. Goodwin, W.J. Grzymala-Busse, X. Zheng, An approach to imbalanced data sets based on changing rule strength, in: S.K. Pal, L. Polkowski, A. Skowron (Eds.), Rough-Neural Computing: Techniques for Computing with Words, Springer, Heidelberg, 2004, pp. 543–553.
[35] D.W. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1) (1991) 37–66, https://doi.org/10.1023/A:1022689900470.
[36] J. Stefanowski, Rough set based rule induction techniques for classification problems, in: Proceedings of 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT 1998), vol. 1, Verlag Mainz, Aachen, 1998, pp. 109–113.
[37] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, 1993.
[38] E. Frank, I.H. Witten, Generating accurate rule sets without global optimization, in: Proceedings of the 15th International Conference on Machine Learning (ICML 1998), Morgan Kaufmann, San Francisco, CA, 1998, pp. 144–151.
[39] J. Fürnkranz, G. Widmer, Incremental reduced error pruning, in: Proceedings of the 11th International Conference on Machine Learning (ICML 1994), Morgan Kaufmann, San Francisco, CA, 1994, pp. 70–77.
[40] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Trans. Syst. Man Cybern. SMC-2 (3) (1972) 408–421, https://doi.org/10.1109/TSMC.1972.4309137.
[41] G.E.A.P.A. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD Explor. Newsl. 6 (1) (2004) 20–29, https://doi.org/10.1145/1007730.1007735.
[42] A. Fernández, S. del Río, N.V. Chawla, F. Herrera, An insight into imbalanced Big Data classification: outcomes and challenges, Complex Intell. Syst. 3 (2) (2017) 105–120, https://doi.org/10.1007/s40747-017-0037-9.
[43] A. Wojna, Analogy-based reasoning in classifier construction, in: J.F. Peters, A. Skowron (Eds.), Transactions on Rough Sets IV, Springer, Heidelberg, 2005, pp. 277–374.
[44] A. Fernández, S. García, M. Galar, R.C. Prati, B. Krawczyk, F. Herrera, Learning from Imbalanced Data Sets, 1st edition, Springer, Cham, 2018, https://doi.org/10.1007/978-3-319-98074-4.
[45] A. Fernández, V. López, M. Galar, M.J.d. Jesus, F. Herrera, Analysing the classification of imbalanced data-sets with multiple classes: binarization techniques and ad-hoc approaches, Knowl.-Based Syst. 42 (2013) 97–110, https://doi.org/10.1016/j.knosys.2013.01.018.
[46] S. Pal, Rough set and deep learning: some concepts, Acad. Lett. (2021) 1849, https://doi.org/10.20935/AL1849.
[47] S.A. Alex, J.J.V. Nayahi, S. Kaddoura, Deep convolutional neural networks with genetic algorithm-based synthetic minority over-sampling technique for improved imbalanced data classification, Appl. Soft Comput. 156 (2024) 111491, https://doi.org/10.1016/J.ASOC.2024.111491.
[48] M.S. Santos, P.H. Abreu, N. Japkowicz, A. Fernández, J.A.M. Santos, A unifying view of class overlap and imbalance: key concepts, multi-view panorama, and open avenues for research, Inf. Fusion 89 (2023) 228–253, https://doi.org/10.1016/J.INFFUS.2022.08.017.
[49] S. Rezvani, X. Wang, A broad review on class imbalance learning techniques, Appl. Soft Comput. 143 (2023) 110415, https://doi.org/10.1016/J.ASOC.2023.110415.
[50] K. Ghosh, C. Bellinger, R. Corizzo, P. Branco, B. Krawczyk, N. Japkowicz, The class imbalance problem in deep learning, Mach. Learn. 113 (7) (2024) 4845–4901, https://doi.org/10.1007/S10994-022-06268-8.