Chi-Spark-RS: an Spark-built Evolutionary Fuzzy Rule Selection Algorithm in Imbalanced Classification for Big Data Problems

Alberto Fernandez and Eva Almansa and Francisco Herrera
Dept. of Computer Science and Artificial Intelligence
University of Granada, Granada, Spain
Emails: alberto@decsai.ugr.es, eva.m.almansa@gmail.com, herrera@decsai.ugr.es

***Abstract*—The significance and benefits of addressing classification tasks in Big Data applications is beyond any doubt. To do so, learning algorithms must be scalable to cope with such a high volume of data. The most suitable option to reach this objective is by using a MapReduce programming scheme, in which algorithms are automatically executed in a distributed and fault tolerant way. Among different available tools that support this framework, Spark has emerged as a "de facto" solution when using iterative approaches.**

In this work, our goal is to design and implement an Evolutionary Fuzzy Rule Selection algorithm within a Spark environment. To do so, we build different local rule bases within each Map Task that are later optimized by means of a genetic process. With this procedure, we seek to minimize the total number of rules that are gathered by each Reduce task to obtain a compact and accurate Fuzzy Rule Based Classification System. In particular, we set the experimental framework in the scenario of imbalanced classification. Therefore, the final objective will be analyzing the best synergy between the novel Evolutionary Fuzzy Rule Selection algorithm and the solutions applied to cope with skewed class distributions, namely cost-sensitive learning, random under-sampling and random-oversampling.

I. Introduction

We are in a world of Big Data where knowledge is power [1]. The large volume of information from which to extract this novel knowledge implies a strong scalability constraints for standard learning algorithms [2]. To cope with this problem, the distributed and fault-tolerant MapReduce programming framework has emerged to fill this gap in terms of computational time [3]. Among several platforms that implements this scheme, Spark must be stressed as the most appropriate solution for iterative-type algorithms [4].

In this work, we focus on the classification task for Big Data problems, and in particular we study the scenario of imbalanced datasets [5]. This kind of problems are defined by having a skewed class distribution, where minority class instances are commonly more difficult to identify than majority ones. To manage the bias of standard classification algorithms towards the majority class, different kind of solutions have been applied [6]: (1) solutions at the data level that increase or reduce the number of instances of the set of the training set; (2) at the algorithm level, where the components of an algorithm are altered so that the minority class becomes more relevant; and (3) cost-sensitive solutions, which alter the procedure algorithm by assigning a higher cost to the minority class.

Among the different techniques for learning classification systems, Fuzzy Rule Based Classification Systems (FRBCS) [7] are considered an effective approach to model complex problems. In particular, they have shown to be quite effective in the context of Big Data problems [8], mainly due to their flexibility and good coverage of the problem space by means of their Knowledge Base (KB). Additional advantages from the use of fuzzy logic are a proper management of the uncertainties derived not only from the collected data, but also for the application of the algorithm themselves [9]. Finally, the use of linguistic fuzzy labels allows for a simpler aggregation of the local models learned during the MapReduce procedure.

The first FRBCS adapted to the MapReduce scheme for a Big Data environment was the Chi-FRBCS-BigData [10]. An extension of the former approach, known as Chi-FRBCS-BigDataCS was released to address imbalanced classification by means of a cost-sensitive learning within the rule weights computation [11]. Newer models are based on fuzzy decision trees [12], in which the fuzzy entropy is applied for definition of strong fuzzy partitions seeking to provide better contextualization of the classification systems into the problem space.

We observe the need of carrying out an in depth learning of the components of FRBCSs in order to reach a superior performance. In this sense, Evolutionary Fuzzy Systems (EFS) [13] are a valuable tool to obtain the optimal parameters in the building process of an FRBCS, or to tune some of its components. However, being based on the use of an evolutionary procedure, an efficient integration for their application in Big Data problems is not straightforward. Therefore, few works are yet developed in this area of research [14], [15].

This contribution proposes the design of an Evolutionary Fuzzy Rule Selection algorithm that allows an improvement of the Chi-FRBCS-BigData models in terms of both performance and compactness of the final Rule Base (RB). To do so, we will comprise a rule selection procedure within each Map task. The success of this approach is based on the following issues:

1) Local models within the subprocesses of each Map may include outliers examples that derive non-relevant rules that must be discarded.
2) Each Map may generate a high number of rules, many of which can be repeated among different nodes. Giving the Reduce task a lower number of rules will improve

the efficiency of the process, as well as simplifying the computation of the proper rule weights in this aggregation step.

3) When addressing an imbalanced problem, the weights associated to minority class rules are usually lower than those for the majority class. In case rules with the same antecedent but different consequent are merged in the Reduce task, majority class ones are more likely to be finally included into the RB.

As stated at the beginning of this work, to assess the proper efficiency in terms of learning time, we must implement our novel approach within the Spark framework. Being built on top of the Chi-FRBCS-BigData implementations, and taking into account that these were developed in Hadoop [16], the first step is to adapt the FRBCSs into this new programming tool by using RDDs so to avoid a high disk overhead. We have noted this novel approach as Chi Spark implementation with Rule Selection (Chi-Spark-RS), being available at GitHub[1].

Our final objective is to analyze the behavior of the FRBCS learned by Chi-Spark-RS under four different case studies in imbalanced classification: applying the learning procedure directly over the original dataset, using a cost-sensitive learning procedure, and applying both random undersampling (RUS) and random oversampling (ROS) to balance the training set.

To carry out this research, this document is divided as follows. Section II presents the preliminaries on imbalanced classification. Next, Section III introduces the baseline learning algorithms for FRBCS in Big Data. Section IV contains the core of this work, describing our Chi-Spark-RS proposal. Then, Section V contains the experimental study to validate the behavior of this novel algorithm. Finally, Section VI presents the conclusions and future work.

II. Classification with imbalanced datasets

The task of classification in imbalanced domains is defined when the elements of a dataset are unevenly distributed among the classes [5], [6]. The majority class, as a result, overwhelms standard learning classification algorithms skewing their performance towards it. The main reason is due to the use of general metrics, such as the percentage of correctly classified observations. In the case of imbalanced datasets, the effect of the minority class hits to the overall accuracy holds a minimum effect, causing a bias to the coverage of the majority class instances, as it was pointed out previously.

Specific methods must be applied so that traditional classifiers are able to deal with the imbalance between classes. Three different methodologies are commonly followed to cope with this problem [5], [6]: data level solutions that rebalance the training set, algorithmic level solutions that adapt the learning stage towards the minority classes, and cost-sensitive solutions which consider different costs with respect to the class distribution.

Among these methodologies, those based on resampling the dataset are widely used, mainly since they are not linked with a single classifier. At this point, three different schemes can be applied: undersampling of the majority class examples, oversampling of the minority class examples, and hybrid techniques.

The simplest approach, RUS, removes instances from the majority class usually until the class distribution is completely balanced. However, this may imply ignoring significant examples from the training data. On the other hand, ROS makes exact copies of existing minority instances. The hitch here is that this method can increase the likelihood of overfitting, as it tends to strengthen all minority clusters disregard their actual contribution to the problem itself.

III. Fuzzy Rule Based Classification Systems for Big Data

The methodology proposed in this work is settled on the basis of the pioneer fuzzy rule learning algorithms for Big Data, namely Chi-FRBCS-BigData [10] and Chi-FRBCS-BigDataCS [11]. As their names suggest, these method were based on the Chi et al.'s approach [17], adapting their implementation into a MapReduce work-flow. The advantages of using such a linguistic model is having the same rule structure and Data Base (DB) for all the distributed subprocesses, thus simplifying the whole design.

In Chi-FRBCS-BigData the initial dataset is divided into several chunks of information which are then fed to the different Map functions. Afterwards, the obtained results are simply aggregated within the Reduce functions. The whole procedure, which is summarized in Figure 1, consists of the following stages:

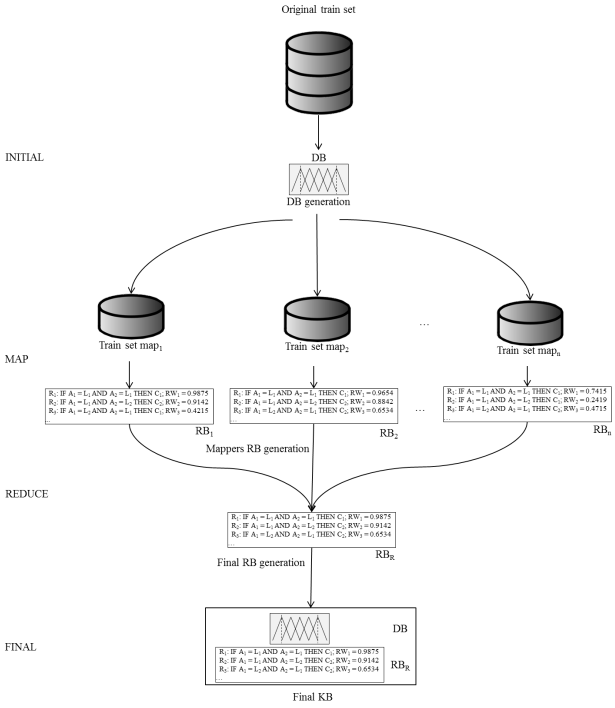

Fig. 1. A flowchart of how the building of the KB is organized in Chi-FRBCS-BigData

[1]https://github.com/aFdezHilario/Chi-Spark-RS

1) *Initial:* the DB is built computing homogeneous fuzzy partitions along the domain of each attribute, depending on the level of granularity selected. Next, the whole training set is divided into independent data blocks which are transferred to the processing units together with the common fuzzy DB.
2) *Map:* In this stage, each processing unit works independently over its available data to build its associated fuzzy RB (called RB_i in Figure 1) following the original Chi-FRBCS method [17]. Specifically, the procedure iterates among all examples, deriving all possible sets of antecedents taking those fuzzy labels that give the highest membership degree per example. To assign a single consequent for each antecedent that was previously obtained, rule weights are computed by means of the Penalized Certainty Factor (PCF) [18] shown in Equation (1), where $\mu_{A_j}(x_p)$ is the membership degree of x_p, i.e. p-th example of the training set with the antecedents of the rule, C_j is the class determined by rule j, and finally γ_p is the cost associated to the class of x_p. This cost is equal to 1 for all classes in Chi-FRBCS-BigData, and $|C_{min}|/|C_p|$ in case of Chi-FRBCS-BigDataCS, with $|C_{min}|$ equal to the number of instances of the minority class. Final class label is set as the one resulting on the greatest Rule Weight (RW).

$$\text{RW}_j = \text{PCF}_j = \frac{\sum\limits_{x_p \in C_j} \mu_{A_j}(x_p) \cdot \gamma_p - \sum\limits_{x_p \notin C_j} \mu_{A_j}(x_p) \cdot \gamma_p}{\sum\limits_{p=1}^{m} \mu_{A_j}(x_p) \cdot \gamma_p} \tag{1}$$

3) *Reduce:* In this third phase, all RB_i computed by a Map process are aggregated to obtain the final RB (called RB_R in Figure 1). As rules with the same antecedent may come from different Maps, we follow the **Chi-FRBCS-BigData-Avg** scheme in which the final RW is computed as the average of those of the rules of the same consequent. In case of having rules with identical antecedent and different consequent, the one with the highest RW is maintained in RB_R.
4) *Final:* results computed in the previous phases are provided as the output of the computation process. The generated fuzzy KB is composed by the fuzzy DB built in the "Initial" phase and the fuzzy RB, RB_R, obtained in the "Reduce" phase. This KB will be the model that will be used to predict the class for new examples.

IV. An Evolutionary Fuzzy System for Rule Selection in Big Data Classification

EFS [13] are models that allow to find the optimal parameters in the building process of an FRBCS or for the a-posteriori improvement of some of its components. Their application to Big Data problems is still at an early stage, as noted by the few works published in this area of research [14], [15]. The time constrains of EFS approaches are probably one of the reasons of the lack of contribution on the topic.

As it was noted at the beginning of this contribution, the scalability in terms of running time is a significant point to evaluate the success of those algorithms for Big Data applications. With this premise, we have developed a MapReduce implementation within the Spark programming framework [4].

In this section, we will describe our EFS proposal based in MapReduce for carrying out rule selection (Subsection IV-A). Then, we will introduce some brief comments regarding the implementation in Spark (Subsection IV-B).

A. Chi-Spark-RS proposal

In this contribution, we propose a methodology for the rule selection of an already learned FRBCS seeking a double aim: 1) simplifying the KB allowing a simpler and more interpretable model; (2) improve the recognition ability of the classifier by means of the synergy among rules with a good cooperation.

The EFS process is applied within each Map task, so that it is locally applied to each RB_i. This scheme seeks to accomplish several objectives:

1) To simplify the whole rule selection process by directly embedding it into each independent Map process.
2) To improve the convergence of the optimization procedure by means of a reduced search space.
3) To provide a better contextualization of the whole process by acting on the data from which rules have been directly derived.
4) To remove "outlier" rules with a high RW that might be erroneously maintained in the final RB during the Reduce step.

In order to apply this tuning, and following the procedure considered in our previous proposals in this topic [15], [19], we will consider the use of the CHC algorithm. The components needed to design this process are explained below:

1) *Coding Scheme:* a binary coding is considered, where each gene represents whether a rule is finally selected (value 1) or not (value 0) from the RB. Therefore, the total number of genes is equal to the size of the initial RB of each Map, i.e. RB_i.
2) *Chromosome Evaluation:* Since we are addressing classification with imbalanced datasets, the fitness function must compensate the uneven class distribution. In order to give the same importance to both the majority and minority class instances, we will make use of the Area Under the ROC curve (AUC) metric [20], computed as:

$$AUC = \frac{1 + TP_{rate} - FP_{rate}}{2} \tag{2}$$

where TP_{rate} is computed as the ratio of correct hits over the minority classes, and FP_{rate} is the ratio of false positives.

Furthermore, we must give a degree of significance on the total number of rules selected, seeking to obtain a simple and compact RB. Hence, the final fitness function is computed as a combination of both factors:

$$Fitness = \alpha \cdot AUC + (1 - \alpha) \cdot RemovedRules \tag{3}$$

3) *Initial Gene Pool:* the initial pool is obtained with the first individual having all genes with value '1' (the initial RB), whereas the remaining individuals are generated at random in $\{0, 1\}$.
4) *Crossover Operator:* We consider the HUX mechanism in which exactly half of the non-matching genes of the chromosomes are swapped between them to create two new offspring.
   Additionally, an incest prevention mechanism is applied prior to the HUX operator. This implies that two parents will be only crossed if they are different enough. This is done by computing their Hamming distance, which must be above a predetermined threshold L, which changes along the genetic process. The initial threshold value is initialized as:

$$L = \#Genes/4.0$$

   where $\#Genes$ stands for the total length of the chromosome (as pointed in item (1)). As stated above, L is decremented by one when there are no new individuals in the next generation

5) *Restarting approach:* When the threshold value is lower than zero, all the chromosomes are regenerated randomly. Furthermore, the best global solution found is included in the population to increase the convergence of the algorithm. Finally, value L is also reset.

B. Spark framework

Currently, there are several programming frameworks that supports MapReduce implementations that allow programs to cope with Big Data problems [3]. Two of the most well-known open-source alternatives are Hadoop [16] and Spark [4]. The advantages of Spark in contrast to Hadoop are clear: applying an in-memory storage of the data to improve the latency between the Map and Reduce tasks, thus allowing a more efficient management of iterative procedures.

To do so, Spark introduces the Resilient Distributed Datasets (RDDs) as well as using two kind of optimized distributed operators over these RDDs: transformations and actions. On the one hand, a transformation can be regarded as an extension of the standard Map process, as it applies a given function over each element of the data and outputs an RDD with the results. On the other hand, an action aggregates all the elements of an RDD and returns the final result to the main program, just as the Reduce task performed. Following this scheme, we may apply an action Reduce over a dataset created from a Map, and it will return only the result of the Reduce execution, instead of the mapped dataset, which is larger. This design allows Spark applications to be executed more efficiently.

We must highlight that, apart from the use of the RDDs and the Spark operators, there are no significant changes in the original Chi-FRBCS-BigData procedure in this Spark implementation. In this way, the standard implementation (without rule selection) has been noted as **Chi-Spark**.

The whole Spark procedure that learns each RB_i, selects the optimal number of rules from each Map by means of the evolutionary search, and aggregates all the rules into a single RB, is noted as **Chi-Spark-CS**. We must also stress that the Scala source code has been made available at GitHub[2].

V. Experimental Study

This section is devoted to show the goodness of our novel algorithm Chi-Spark-RS with respect to the standard learning procedure without rule selection. In addition to the former, we will analyze the synergy between this approach and those solutions for addressing imbalanced classification, namely cost-sensitive learning, RUS and ROS.

To do so, we will first introduce our experimental framework in which the details of the benchmark datasets and the parameters are given (Subsection V-A). Then, we will show our experimental results and we will carry out the subsequent analysis (Subsection V-B).

A. Experimental Framework

For this study, we have selected three Big Data problems from UCI repository [21]: Covertype, Poker, and KddCup99 datasets. These problems are translated into four binary datasets by joining pairs of classes or contrasting one class versus the rest. A summary of the problem features is shown in Table I, where the number of examples (#Ex.), number of attributes (#Atts.), selected classes, number of examples per class, and the Imbalance Ratio (IR) are included. Table is in descending order according to the number of examples.

TABLE I
Summary of BigData classification problems

Datasets	#Ex.	#Atts.	Selected classes	#Samples per class	IR
Poker_0_vs_5	515,751	10	(0; 5)	(513,701; 2,050)	250.59
Poker_0_vs_2	562,529	10	(0; 2)	(513,701; 48,828)	10.52
Covtype_7_vs_all	581,012	54	(7; remainder)	(560,445; 20,567)	27.25
Kddcup_R2L_vs_all	4,898,431	41	(R2L; remainder)	(4,897,305; 1,126)	437.73

For the experimental analysis, we will take into account the *AUC metric* to evaluate the classification performance in the context of imbalanced datasets. The estimates for this metric will be obtained by means a 5-fold stratified cross-validation partitioning scheme.

The configuration parameters for Chi-Spark (the Spark implementation of Chi-FRBCS-BigData) and Chi-Spark-RS (our novel proposal) are presented in Table II being "Conjunction operator" the operator used to compute the compatibility degree of the example with the antecedent of the rule and the operator used to compute the compatibility degree and the RW. We must recall that regarding the "Reduce" stage we will make use of the *Chi-FRBCS-BigData-Avg* version, as stated in Section III.

Regarding the infrastructure used to perform the experiments, we have used the research group's cluster with 16 nodes connected with a 40Gb/s Infiniband. Each node is equipped with two Intel E5-2620 microprocessors (at 2 GHz,

[2]https://github.com/aFdezHilario/Chi-Spark-RS

TABLE II
CONFIGURATION PARAMETERS FOR CHI-SPARK AND CHI-SPARK-RS

Number of Labels:	3 fuzzy partitions
Conjunction operator:	Product T-norm
Rule Weight:	Penalized Certainty Factor [18]
Fuzzy Reasoning Method:	Winning Rule
α (for fitness)	0.7 for AUC
Number of evaluations (CHC)	1,000
Population Size (CHC)	50 individuals

15MB cache) and 64GB of main memory running under Linux CentOS 6.6. The head node of the cluster is equipped with two Intel E5-2620 microprocessors (at 2 GHz, 15MB cache) and 96GB of main memory. Furthermore, the cluster works with Hadoop 2.6.0 (Cloudera CDH5.4.2). As total number of Maps for the data distribution we have selected 128 Maps.

Finally, three different mechanisms to address imbalanced classification will be applied to evaluate their impact in synergy with Chi-Spark-RS: Cost-sensitive learning via Chi-FRBCS-BigDataCS, RUS and ROS. These last algorithms are available in a Spark package called Imb-sampling-ROS_and_RUS[3]. The datasets obtained by the pre-processing were generated to achieve a balanced class distribution.

B. Analysis of the results

In this section, we will analyze the behavior of Chi-Spark-RS under three different perspectives: (1) global performance (in terms of AUC), interpretability of the RB (in terms of number of rules), and running time. The objective is to contrast whether there are significant differences among these metrics depending on the solution applied to address imbalance.

To do so, we show in Tables III to VI the complete experimental results. Each table contains the train and test AUC values, the number of rules, and the time (hh:mm:ss) for each one of the four datasets selected for this study. All tables are divided into two parts; the left-hand side corresponds to the results obtained by the standard fuzzy rule learning approach, i.e. Chi-Spark, whereas the right-hand side shows the results for our new proposed Chi-Spark-RS. Finally, we must point out that each one of the rows corresponds to a different case study for the solutions applied to cope with imbalanced classification.

From the results we may extract the following conclusions:

- Best AUC results are obtained using RUS and ROS prior to the learning stage. This justifies the necessity of balancing the class distribution so that general classifiers are able to correctly identify both classes of the problem.
- We must stress the good behavior shown by RUS in both Chi-Spark and Chi-Spark-RS. As the number of examples is significantly reduced from the training set, especially in the highly imbalanced problems, both the number of generated rules and the training time are also decreased in contrast to the remaining case studies.
- Finally, we must stress the reduction rate in terms of number of rules achieved by Chi-Spark-RS. In all cases it gets around a 20% of less rules while maintaining or improving the AUC metric. As stated in the previous point, the most interesting behavior is shown in synergy with RUS preprocessing, where a very low number of rules (in proportion to the original number of examples) is enough to represent the problem accurately.

[3]https://spark-packages.org/package/saradelrio/Imb-sampling-ROS_and_RUS

VI. CONCLUDING REMARKS

In this work we have carried out a new Evolutionary Fuzzy Rule Selection proposal for classification with Big Data problems. The idea behind this research was to enhance the rule generation mechanism of a MapReduce learning algorithm, by obtaining an optimal number of rules within each Map process. To do so, we have implemented the whole procedure within Spark, aiming for an efficient solution.

In particular, we have analyzed and contrasted the behavior of this approach in the context of imbalanced classification. To do so, four different case studies have been set: using the original dataset, applying a cost-sensitive learning scheme, rebalancing the training set via RUS, and rebalancing the training set via ROS. In the two first cases, the global performance has maintained similar to the original Chi-Spark approach, but the RB reduction must be emphasized. When applying sampling to the dataset, we have observed an improvement of the results with respect to the previous case. A very interesting behavior must be highlighted in the case of RUS, where the size of the problems have been significantly reduced but still the fuzzy learning approach has obtained a good performance for both the majority and minority classes. This behavior must be related to the coverage of fuzzy labels, where just few "prototype" examples are needed to derive rules that provide a good representation of the problem space.

Results have been very promising, but there is still much work to be carried out for future research. In particular, two extensions can be directly considered. First, to avoid the inclusion in the final RB of those rules that have been marked to be removed in the majority of the Map processes. Second, to apply a global Evolutionary Rule Selection after the whole generation of the RB.

ACKNOWLEDGMENT

This work have been partially supported by the Spanish Ministry of Science and Technology under projects TIN2014-57251-P and TIN2015-68454-R.

REFERENCES

[1] A. Labrinidis and H. V. Jagadish, "Challenges and opportunities with big data," *VLDB*, vol. 5, no. 12, pp. 2032–2033, 2012.

[2] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, 2014.

[3] A. Fernández, S. Río, V. López, A. Bawakid, M. del Jesus, J. Benítez, and F. Herrera, "Big data with cloud computing: An insight on the computing environment, mapreduce and programming framework," *WIREs Data Mining and Knowl. Discovery*, vol. 4, no. 5, pp. 380–409, 2014.

[4] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12, 2012, pp. 1–14.

TABLE III
EXPERIMENTAL RESULTS FOR POKER_0_VS_5 DATASET. BEST RESULT PER METRIC IS STRESS IN BOLDFACE.

Method	AUC-Tr	AUC-Tst	#Rules	Time	Method	AUC-Tr	AUC-Tst	#Rules	Time
Chi-Spark	.7602	.7555	49013	00:00:08	Chi-Spark-RS	.7677	.7642	42498	00:01:17
Chi-Spark-CS	.7693	.7644	49013	00:00:10	Chi-Spark-CS-RS	.7782	.7708	42515	00:01:25
RUS+Chi-Spark	.9834	.8460	2013	**00:00:03**	RUS+Chi-Spark-RS	.7950	.7120	**728**	**00:00:03**
ROS+Chi-Spark	.7413	.7412	49013	00:00:11	ROS+Chi-Spark-RS	.9461	**.9371**	42574	00:05:15

TABLE IV
EXPERIMENTAL RESULTS FOR POKER_0_VS_2 DATASET. BEST RESULT PER METRIC IS STRESS IN BOLDFACE.

Method	AUC-Tr	AUC-Tst	#Rules	Time	Method	AUC-Tr	AUC-Tst	#Rules	Time
Chi-Spark	.6714	.6295	51087	00:00:09	Chi-Spark-RS	.6567	.6174	44463	00:01:38
Chi-Spark-CS	.6719	.6298	51087	00:00:09	Chi-Spark-CS-RS	.6597	.6201	44532	00:02:01
RUS+Chi-Spark	.7475	**.6483**	30896	**00:00:05**	RUS+Chi-Spark-RS	.7125	.6349	**20259**	00:00:23
ROS+Chi-Spark	.5779	.5608	51087	00:00:10	ROS+Chi-Spark-RS	.7036	.6406	44404	00:05:03

TABLE V
EXPERIMENTAL RESULTS FOR COVTYPE_7_VS_ALL DATASET. BEST RESULT PER METRIC IS STRESS IN BOLDFACE.

Method	AUC-Tr	AUC-Tst	#Rules	Time	Method	AUC-Tr	AUC-Tst	#Rules	Time
Chi-Spark	.8563	.8484	8126	00:00:07	Chi-Spark-RS	.8522	.8466	7296	00:02:28
Chi-Spark-CS	.8574	.8500	8126	00:00:08	Chi-Spark-CS-RS	.8569	.8499	7271	00:02:41
RUS+Chi-Spark	.9383	**.9277**	4037	**00:00:05**	RUS+Chi-Spark-RS	.9337	.9253	**2951**	00:00:13
ROS+Chi-Spark	.7085	.7058	8126	00:00:08	ROS+Chi-Spark-RS	.9148	.9068	7314	00:05:15

TABLE VI
EXPERIMENTAL RESULTS FOR KDDCUP_R2L_VS_ALL DATASET. BEST RESULT PER METRIC IS STRESS IN BOLDFACE.

Method	AUC-Tr	AUC-Tst	#Rules	Time	Method	AUC-Tr	AUC-Tst	#Rules	Time
Chi-Spark	.6112	.6047	1152	00:00:05	Chi-Spark-RS	.5978	.5946	903	00:19:33
Chi-Spark-CS	.6064	.6047	1152	00:00:07	Chi-Spark-CS-RS	.6203	.6179	867	00:20:11
RUS+Chi-Spark	.9325	.9261	131	**00:00:01**	RUS+Chi-Spark-RS	.9845	**.9761**	**40**	00:00:08
ROS+Chi-Spark	.5347	.5329	1152	00:01:21	ROS+Chi-Spark-RS	.9679	.9588	885	01:00:11

[5] V. Lopez, A. Fernandez, S. Garcia, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, no. 20, pp. 113–141, 2013.

[6] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Syst. with Applicat.*, vol. 73, pp. 220 – 239, 2017.

[7] H. Ishibuchi, T. Nakashima, and M. Nii, *Classification and modeling with linguistic information granules: Advanced approaches to linguistic data mining*. Berlin, Germany: Springer-Verlag, 2004.

[8] A. Fernandez, C. Carmona, M. del Jesus, and F. Herrera, "A view on fuzzy systems for big data: Progress and opportunities," *Int. J. of Computational Intell. Syst.*, vol. 9, no. 1, pp. 69–80, 2016.

[9] H. Wang, Z. Xu, and W. Pedrycz, "An overview on the roles of fuzzy set techniques in big data processing: Trends, challenges and opportunities," *Knowledge-Based Syst.*, vol. 118, pp. 15–30, 2017.

[10] S. Río, V. López, J. Benítez, and F. Herrera, "A mapreduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules," *International Journal of Computational Intelligence Systems*, vol. 8, no. 3, pp. 422–437, 2015.

[11] V. López, S. del Río, J. M. Benítez, and F. Herrera, "Cost-sensitive linguistic fuzzy rule based classification systems under the mapreduce framework for imbalanced big data," *Fuzzy Sets and Systems*, vol. 258, pp. 5–38, 2015.

[12] A. Segatori, F. Marcelloni, and W. Pedrycz, "On distributed fuzzy decision trees for big data," *IEEE Trans. Fuzzy Syst.*, vol. PP, no. 99, pp. 1–1, 2017.

[13] A. Fernandez, V. Lopez, M. J. del Jesus, and F. Herrera, "Revisiting evolutionary fuzzy systems: Taxonomy, applications, new trends and challenges," *Knowlegde Based Systems*, vol. 80, pp. 109–121, 2015.

[14] A. Ferranti, F. Marcelloni, and A. Segatori, "A multi-objective evolutionary fuzzy system for big data," in *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2016, pp. 1562–1569.

[15] A. Fernndez, S. del Ro, and F. Herrera, "A first approach in evolutionary fuzzy systems based on the lateral tuning of the linguistic labels for big data classification," in *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2016, pp. 1437–1444.

[16] C. Lam, *Hadoop in action*, 1st ed. Manning, 2011.

[17] Z. Chi, H. Yan, and T. Pham, *Fuzzy algorithms with applications to image processing and pattern recognition*. World Scientific, 1996.

[18] H. Ishibuchi and T. Yamamoto, "Rule weight specification in fuzzy rule-based classification systems," *IEEE Trans. Fuzzy Syst.*, vol. 13, pp. 428–435, 2005.

[19] A. Fernández, M. J. del Jesus, and F. Herrera, "On the 2–tuples based genetic tuning performance for fuzzy rule based classification systems in imbalanced data–sets," *Inform. Sci.*, vol. 180, no. 8, pp. 1268–1291, 2010.

[20] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 3, pp. 299–310, 2005.

[21] M. Lichman, "UCI machine learning repository; university of california, irvine, school of information and computer sciences. http://archive.ics.uci.edu/ml," 2013. [Online]. Available: http://archive.ics.uci.edu/ml