A Meta-Learning Method to Select Under-Sampling Algorithms for Imbalanced Data Sets

Romero F. A. B. de Morais, Péricles B. C. Miranda and Ricardo M. A. Silva
Universidade Federal de Pernambuco
Pernambuco, Recife

Abstract—**Imbalanced data sets originating from real world problems, such as medical diagnosis, can be found pervasive. Learning from imbalanced data sets poses its own challenges, as common classifiers assume a balanced distribution of examples' classes in the data. Sampling techniques overcome the imbalance in the data by modifying the examples' classes distribution. Unfortunately, selecting a sampling technique together with its parameters is still an open problem. Current solutions include the brute-force approach (try as many techniques as possible), and the random search approach (choose the most appropriate from a random subset of techniques). In this work, we propose a new method to select sampling techniques for imbalanced data sets. It uses Meta-Learning and works by recommending a technique for an imbalanced data set based on solutions to previous problems. Our experimentation compared the proposed method against the brute-force approach, all techniques with their default parameters, and the random search approach. The results of our experimentation show that the proposed method is comparable to the brute-force approach, outperforms the techniques with their default parameters most of the time, and always surpasses the random search approach.**

Index Terms—**Meta-learning; Algorithm selection; Sampling algorithms;**

I. Introduction

Machine learning applications can range from classification of modules in software as faulty to detection of cancer in patients [1], [2]. What many of those applications have in common is an imbalanced distribution of classes among their examples. For instance, past data about patients that underwent tests to discover whether they had cancer, are expected to have a low ratio of cancerous to non-cancerous patients. Common classifiers assume an even distribution of classes in the data, an assumption that if not met may hinder learning [3]. Current solutions to the problem of learning from imbalanced data mainly fall into two categories: modifying the distribution of classes in the data, and adapting existing classifiers' algorithms to deal with the imbalance.

Given an imbalanced data set, sampling techniques can be employed to generate a new data set where the distribution of examples in each class is more balanced. Over-sampling either adds repeated examples from the minority class or synthesizes from the available data. On the other hand, under-sampling simply removes examples from the majority class. Even though under-sampling techniques discard information about the majority class, the loss may be compensated by the reduction in necessary computational resources (less time to train the models and less space to store the data). Moreover, noisy and redundant instances when removed may even contribute to a more robust data set. Given the aforementioned reasons, under-sampling techniques have become largely used in the context of imbalanced data sets [3].

The process of selecting a sampling technique and its parameters to pre-process an imbalanced data set is still problematic, with few solutions existing for it. A brute-force search through all combinations of algorithms and their parameters can find the best solution, as performed in [4], [5]. Nonetheless, due to time constraints, a brute-force search is not always feasible. Instead of using the brute-force approach, Tong et al. [6] conceived an analytical method to find an optimal configuration for a combination of the random over-sampling and the random under-sampling techniques. Although an analytical solution was devised, the method still had to evaluate a number of configurations before selecting the most appropriate configuration. Another possibility is to choose the most suitable solution from a random subset of algorithms and configurations (i.e., perform a random search).

In this work, we propose a novel way of selecting a sampling algorithm and its parameters for an imbalanced data set. The method relies on Meta-Learning (MtL), and it works by recommending a solution to a new problem based on stored solutions for previous problems (meta-data). In greater detail, given a new imbalanced data set (which we call a problem here), the most similar problem is retrieved from the meta-data and its solution is recommended. MtL is a less expensive solution when compared to the brute-force approach and search methods. Once the knowledge is obtained by the meta-learner, algorithms can be recommended for new problems without the necessity of empirically assessing different candidate configurations as performed using search techniques [7]. It is worth mentioning that the MtL method has not been investigated for the current problem, and the optimization of under-sampling algorithms for imbalanced data sets is a task that needs further investigation.

Seven well-known under-sampling algorithms were considered here for evaluation of the proposed method. For each algorithm a range of parameters' values were defined, totaling 491 possible variations of algorithms that can be recommended for an input imbalanced problem. The proposed solution was evaluated on 29 different classification problems and compared to a brute-force approach, the random search method, and also compared with the seven under-sampling algorithms using their default values. The results showed that the recommendation always surpassed the random search, outperformed the algorithms using their default parameters most of the time, and it was usually comparable to the brute-force solution.

This work is organized as follows. Section II presents other works related to the task of selecting/optimizing sampling algorithms for imbalanced data sets. Section III describes the proposed MtL method. Section IV brings the experimental methodology and presents the obtained results. Finally, in Section V, the conclusions and possible future works are presented.

II. Related Work

In this section, we briefly discuss the references that tried to improve the process of pre-processing an imbalanced data set via sampling techniques. This is achieved either through the optimization of the sampling techniques parameters or recommendation of a method.

In [4] Hulse et al. conduct a comprehensive study of the impact of seven sampling techniques on 11 different classifiers trained on 35 data sets. In their work, only three of the seven sampling techniques were under-sampling techniques. Since several values for the sampling parameters were utilized, a total of 31 combinations of sampling techniques plus parameters were employed in their study. A brute-force approach was used to determine the best combinations of classifiers and sampling techniques for each data set, resulting in over a million classifiers trained.

Cieslak and Chawla [5] believe that data are mostly multimodal and sampling techniques should be applied locally rather than globally. Hence, they devised a method that creates a partition of the data, and for each member of the partition it determines the best combination of sampling technique plus its parameters. Again, a brute-force approach is used to select the best parameters and the results of the proposed technique (called Local Sampling) are compared to another three sampling techniques (one under-sampling and two over-sampling techniques), achieving better results.

Instead of applying the brute-force approach to find the best parameters for the random under-sampling and the random over-sampling techniques, Tong et al. [6] conceived an analytical method to find optimal levels of random under-sampling and random over-sampling for an imbalanced data set. It employs Design of Experiments (DOE) and Response Surface Methodology (RSM) to obtain the optimal levels of sampling. Although an analytical solution was devised, in order to construct the RSM model, several runs of a cross-validated procedure have to be executed to obtain mean values of the response variable (e.g. AUC) for various levels of sampling. The new method (called S-RSM) was compared to the default (set to balance the classes' distribution) random under-sampling and random over-sampling techniques, being superior to both methods alone.

Recently, Loyola-González et al. [8] investigated the impact of sampling techniques on contrast pattern classifiers. Their experimentation included 20 sampling techniques (nine over-sampling, eight under-sampling, and three hybrid methods), two contrast pattern classifiers, and 95 imbalanced data sets. The data sets were grouped according to their imbalance ratio (number of majority examples / number of minority examples), and based on their results they suggest a guide for selecting sampling techniques given the imbalance ratio of an imbalanced data set. In their future works section, they plan to use data set intrinsic characteristics to select sampling techniques instead of using the imbalance ratio.

The brute-force and the random search approaches comprise most of the attempts to solve the problem at hand. However, it is known that there is a vast number of combinations of sampling algorithms and possible values for their hyper-parameters. Thus, the application of these approaches on the optimization of under-sampling algorithms may become impracticable.

In the next section, we present the proposed method which aims to provide an optimized under-sampling algorithm, for an input imbalanced data set, in a less expensive way.

III. Proposal

This work proposes an automatic methodology for recommendation of an adequate under-sampling algorithm for an input imbalanced problem. Meta-Learning handles the under-sampling algorithm selection for imbalanced problems as a supervised learning task. Figure 1 presents the architecture of the proposed method. Each training example for MtL, known as a meta-example, is composed of the characteristics of a past imbalanced problem and information regarding the performance achieved by a set of candidate under-sampling algorithms on the problem. By using, as input, a set of such meta-examples, a meta-learner can predict the most suitable algorithm for a new problem based on its characteristics. MtL is a less expensive solution when compared to the brute-force approach and search methods [7]. In fact, once the knowledge is obtained by the meta-learner, algorithms can be recommended for new problems without the necessity of empirically assessing different candidate configurations as performed using search techniques.

The MtL methodology includes (1) the generation of meta-data regarding the performance of a set of under-sampling algorithms on existing imbalanced data sets, and (2) the use of the generated meta-data to predict an algorithm to pre-process a new imbalanced data set. Both points will be described in detail in the following sections.

A. Generation of meta-data

To obtain the meta-data, two types of operations are necessary: i) execute the experiments on the available data sets with the available algorithms and selected parameter settings, ii) calculate a set of meta-features describing those data sets. These procedures will be explained as follows.

Defining candidate algorithms: In this work, we adopted seven well-known under-sampling algorithms to be considered in the recommendation process. The name of each under-sampling technique, together with a short description and a list of its parameters is presented in Table I. As each parameter of

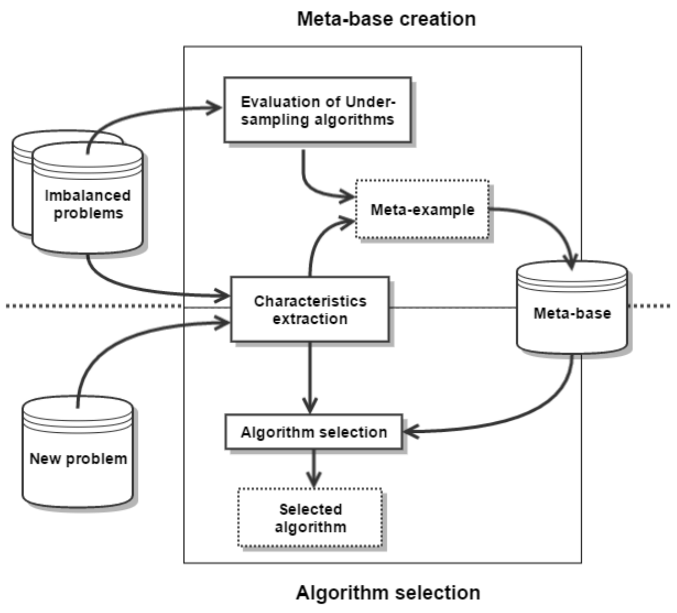

Fig. 1. Architecture of the proposed method.

each algorithm assumes several possible values, the number of possible algorithm configurations becomes large. As we cannot compute the performance of all possible algorithms, a subset needs to be determined. The number of choices was defined considering the trade-off between the quality of results and computational requirements. On one hand, it is interesting to have a large number of algorithms, allowing us to identify, in every case, one that is close to the best possible algorithm. On the other hand, if too many alternatives were admitted, the computational demands to run all associated experiments and collect the meta-data required for meta-learning could be too high.

All techniques that use the k-NN algorithm (whose default value is $k = 3$), adopted $[2, 15]$ as the range of k values. From this range, we have selected 14 alternatives, following an arithmetic progression with a factor of 1. Regarding the KMUS technique, which uses the k-means algorithm (whose default value k is equals to the size of the minority class), it adopted positive multiples of the size of the minority class as the interval of values (up to 5 times). From this range, we have selected 5 alternatives, following an arithmetic progression with a factor of 1. Those techniques that have the m hyper-parameter (whose default value is the percentage value where both majority and minority classes have the same number of examples), adopted $[0, 80]$ as the interval of values, where the value 0 means the default behavior. From this range, we have selected 9 alternatives, following an arithmetic progression with a factor of 10. Those techniques that have the p hyper-parameter (whose default value is $p = 2$), adopted the interval $[1, 5]$. From this range, we have selected 5 alternatives, following an arithmetic progression with a factor of 1. This totaled 491 candidate algorithms that can be recommended to an input imbalanced problem. It is worth mentioning that all parameters' ranges previously outlined were defined empirically.

Collecting performance data: To evaluate the quality of a single combination of algorithm and parameters, a ten-fold cross-validated resampling procedure repeated five times was employed. For classification purposes, a Support Vector Machine (SVM) with a Gaussian Kernel was adopted. The SVM's parameters were held constant. C had a fixed value of 1 throughout all problems, while σ had a fixed value computed for each problem following the heuristic defined in [12]. When dealing with imbalanced data sets, performance of a classifier cannot be assessed solely based on accuracy. In a problem where 99% of the examples belong to one class, predicting all the examples to be part of the majority class would yield an accuracy of 99%, which is clearly undesirable. Hence, we created a weighted performance metric (WPC) to assess a classifier's performance:

$$\begin{aligned} WPC = 0.05 \times Acc + 0.25 \times AUC + 0.40 \\ \times F_1 + 0.15 \times Spec + 0.15 \times NPV. \end{aligned} \tag{1}$$

Where Acc stands for Accuracy, AUC for Area Under the ROC Curve, F_1 for the F_1 score, $Spec$ for Specificity, and NPV for Negative Predictive Value. NPV is the analog of Precision for the negative (majority) class. The metric ranges from 0 to 1, as it is a convex combination of other metrics that also vary from 0 to 1. The objective is to maximize the WPC.

The weights for the WPC metric were chosen empirically. Recognition of the minority class was considered the most important factor and a weight of 0.40 was given to the F_1 score. Since the AUC is used in many previous works [4], [5], [6], [8], it was given a weight of 0.25. Although recognition of the majority class is usually high in imbalanced problems, in response to this, a weight of 0.15 was given to $Spec$ and NPV each. Finally, as Acc is not a suitable metric for imbalanced problems, but widely adopted in the Machine Learning community, it was given a weight of 0.05.

For the purpose of algorithm selection, a proper set of meta-features, i.e., data set characteristics, need to satisfy the following two conditions. First, it must extract useful information to determine the relative performance of the individual learning algorithms. Secondly, calculating them should not be too costly. In other words, calculating the measures should be cheaper than running the individual candidate algorithms, otherwise, one might simply execute them as well. In this work, we have used the meta-features described in [13], which are listed in Table II.

B. Prediction with the k-NN ranking method

The use of the k-NN ranking method to provide a recommendation of under-sampling algorithms for imbalanced problems includes the following steps. First, calculating the meta-features for the data set in question. Second, identifying the k nearest neighbors among the existing data sets. Third, retrieving the rankings of algorithms on the nearest neighbors and use this information to make the recommended ranking.

TABLE I
THE NAME OF EACH UNDER-SAMPLING TECHNIQUE CONSIDERED IN THIS WORK, TOGETHER WITH A SHORT DESCRIPTION AND A LIST OF ITS PARAMETERS.

Under-sampling Techniques		
Technique Name	Short Description	Hyper-parameters
Edited Nearest Neighbors (ENN or Wilson's Editing) [9]	For each majority class example, find its *k*-nearest neighbors and classify it accordingly to them, breaking ties randomly. If it is misclassified, remove it.	*k*: Number of neighbors.
k-means Under-sampling (KMUS)	Run a *k*-means algorithm on the majority class examples and select the *k* centroids to represent it.	*k*: Number of centroids.
Most Distant (MD) [10]	For each majority class example, compute its average distance to the *k*-nearest minority class neighbors. Select *m*% of the most distant majority examples.	*k*: Number of neighbors. *m*: Percent of examples to select from the majority class.
Neighborhood Cleaning Rule (NCL) [11]	For each example, find its *k*-nearest neighbors and classify it accordingly to them, breaking ties randomly. For every misclassified example, either remove it if it belongs to the majority class or remove its majority neighbors if it belongs to the minority class.	*k*: Number of neighbors.
NearMiss-1 (NM1) [10]	For each majority class example, compute its average distance to the *k*-nearest minority class neighbors. Select *m*% of the closest majority examples.	*k*: Number of neighbors. *m*: Percent of examples to select from the majority class.
NearMiss-3 (NM3) [10]	For each minority class example, compute its *k*-nearest majority class neighbors and select $p \leq k$ of them to represent the majority class.	*k*: Number of neighbors. *p*: Number of majority class examples to select from the *k* neighbors.
Random Under-sampling (RUS)	Randomly selects *m*% examples from the majority class.	*m*: Percent of examples to select from the majority class.

TABLE II
MEASURES USED TO CHARACTERIZE DATA SETS (META-FEATURES).

Simple
Number of examples
Number of attributes
Ratio of the number of examples to the number of attributes
Proportion of the attributes with outliers
Presence of outliers in the target
Statistical
Coefficient of variation of the target (ratio of the standard deviation to the mean)
Sparsity of the target (coefficient of variation discretized into three values)
Stationarity of the target (the standard deviation is larger than the mean)
Average absolute correlation between numeric predictor attributes
Average dispersion gain

The meta-features represent data characteristics of the data set presented earlier. To determine the k nearest neighbors, the distance between the meta-features of the data set in question and all the others is calculated, and the k closest data sets are selected. The distance function (dist) implemented was the Euclidean distance, defined as:

$$dist(a, b) = \sqrt{\sum_{i=1}^{p} (a_i - b_i)^2} \tag{2}$$

where $a = (a_1, a_2, ..., a_p)$ and $b = (b_1, b_2, ..., b_p)$ are the vectors of meta-features.

IV. EXPERIMENTS AND RESULTS

In this section, we present the experiments that evaluated the proposed solution on the set of 29 classification problems (see column one of the Table III), totaling 29 meta-examples (one for each problem). The 29 data sets were collected from three different sources: the PROMISE repository [14], the KEEL-data set repository [15], and the UCI Machine Learning repository [16]. The employed data sets from the PROMISE repository have been cleaned by Shepperd et al.

[17]. For data gathered from the KEEL-data repository and the UCI Machine Learning repository, we cleaned them by first removing repeated instances, followed by removal of inconsistent instances (same values for the predictors but different class).

All the under-sampling algorithms were implemented under the R environment [18]. The model training process, including the cross-validated resampling procedure, was conducted using the caret [19], and the kernlab (for the SVM) packages [12]. The MtL framework was implemented in Python, and the communication with the R code was established via the rpy2 package [20].

The proposed solution was evaluated by following a leave-one-out cross-validated (loocv) resampling procedure. At each step of the leave-one-out procedure, one meta-example was left out (considered as input problem) to evaluate the implemented prototype, and the remaining 28 meta-examples were considered in the meta-data to be selected by the meta-learner. Thus, the meta-learner recommend, for the input problem, the best algorithm (with highest WPC) adopted to solve the most similar (we used the k-NN with $k = 1$) meta-example regarding the input problem. To assess the performance of the proposal's recommendation, the recommended algorithm is executed on the input problem and its performance is evaluated using the WPC metric, presented in Equation 1.

Table III presents the WPC values of the proposal and other three approaches. The first column, from the *Algorithms* area, presents the average WPC values achieved by the best algorithms found by a brute-force approach for each problem. In other words, the results presented in this column represent the best possible WPC values for each problem.

The second column presents the WPC values achieved by the proposed approach. The third column, named *Default*, presents the best WPC value reached by the seven algorithms when configured with their default parameters. Finally, the fourth column presents the WPC values reached by a random search approach using 50 iterations. Both brute-force and random search algorithms are implementations from the Scikit Learn library [21].

The values, presented in Table III, which are in bold mean that the values are equal to the results reached by the brute-force approach; and the values which have the symbol (▼) mean that these values were overcome by the counterparts. As it can be seen in Table III, the results achieved by the algorithms recommended by the proposal overcame all results reached by the random search approach. Besides, the proposal overcame the algorithms using default parameters in 23 classification problems, lost in only 4 and drew in 2 problems. In general, the results obtained by the proposal were close to the global optimum, and in 8 out of the 29 problems the global optimum was reached.

The results show that the task of choosing an adequate under-sampling algorithm for an imbalanced problem can be solved using the proposed approach. The proposal reduces the human intervention on this task, automatically providing an adequate algorithm for the input problem.

V. Conclusions

In this work, a novel methodology to select a sampling technique for an imbalanced data set was proposed. The methodology not only addresses the problem of selecting a sampling technique but also of defining its parameters. It works by recommending a sampling technique for an imbalanced data set according to solutions that worked well for previous problems (the meta-data).

The results of our experimentation showed that our method was comparable to the brute-force approach, surpassed the sampling algorithms with their default parameters most of the time, and always outperformed the random search.

Once the meta-data is built, the proposed method has the advantage of finding a solution in linear-time (on the size of the meta-data). Disadvantages to the proposed method include the time to build the meta-data, and the fact that solutions that were never optimal for any previous problem will never be recommended. Hence, the necessity of building a large meta-data from the outset.

Here, we considered only under-sampling techniques to assess the proposed method. However, the methodology could be easily extended to include over-sampling and hybrid techniques.

Future work include the refinement of the recommendation process. Instead of recommending a single solution based on the most similar problem in the meta-data, the system could recommend a combination (or sequence) of techniques to be applied to a new problem.

Acknowledgment

The authors would like to thank Motorola Mobility for funding this research.

References

[1] K. S. Woods, C. C. Doss, K. W. Bowyer, J. L. Solka, C. E. Priebe, and W. P. Kegelmeyer JR, "Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 06, pp. 1417–1436, 1993.

[2] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," *Reliability, IEEE Transactions on*, vol. 62, no. 2, pp. 434–443, 2013.

[3] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. on Knowl. and Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2008.239

[4] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "Experimental perspectives on learning from imbalanced data," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 935–942.

[5] D. A. Cieslak and N. V. Chawla, "Start globally, optimize locally, predict globally: Improving performance on imbalanced data," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 143–152.

[6] L.-I. Tong, Y.-C. Chang, and S.-H. Lin, "Determining the optimal re-sampling strategy for a classification model with imbalanced data using design of experiments and response surface methodologies," *Expert Systems with Applications*, vol. 38, no. 4, pp. 4222–4227, 2011.

[7] C. Soares, P. B. Brazdil, and P. Kuba, "A meta-learning method to select the kernel width in support vector regression," *Machine learning*, vol. 54, no. 3, pp. 195–209, 2004.

TABLE III
WPC VALUES REACHED BY THE PROPOSAL AND COUNTERPARTS.

Problem Name	**Algorithms**			
	Brute-Force	Proposal	Default	Random Search
banknote authentication	1.0000	**1.0000**	**1.0000**	0.9700 ▼
Blood Transfusion Service Center	0.7393	0.7273	0.7075 ▼	0.6400 ▼
Breast Cancer Wisconsin (Original)	0.9568	0.9524	0.9164 ▼	0.8900 ▼
cleveland-0_vs_4	0.9481	0.8833	0.8541 ▼	0.7200 ▼
Diabetic Retinopathy Debrecen	0.7288	0.7106 ▼	**0.7288**	0.6500 ▼
ecoli1	0.8962	**0.8962**	0.8620 ▼	0.7800 ▼
glass1	0.7707	0.7685	0.7441 ▼	0.6900 ▼
haberman	0.6888	**0.6888**	0.6544 ▼	0.5900 ▼
iris0	1.0000	0.9940 ▼	**1.0000**	0.9200 ▼
JM1”	0.6625	0.6476	0.6124 ▼	0.5700 ▼
KC3”	0.7523	0.6979	0.6230 ▼	0.5500 ▼
MC1”	0.7942	0.6058 ▼	0.7942	0.5200 ▼
MC2”	0.7194	0.6860	0.6199 ▼	0.5500 ▼
MW1”	0.7737	0.7618	0.7475 ▼	0.6500 ▼
new-thyroid1	0.9861	0.9650	0.9417 ▼	0.8900 ▼
page-blocks0	0.9481	0.9264	0.9047 ▼	0.8800 ▼
PC1”	0.7616	**0.7616**	0.7283 ▼	0.6200 ▼
PC2”	0.7319	0.6607	0.6497 ▼	0.5800 ▼
PC3”	0.7379	0.7146	0.6726 ▼	0.6300 ▼
PC4”	0.8301	0.7996	0.7733 ▼	0.6700 ▼
PC5”	0.7164	**0.7164**	0.6853 ▼	0.6400 ▼
pima	0.7711	0.7622	0.7287 ▼	0.7100 ▼
QSAR biodegradation	0.8907	0.8742	0.8128 ▼	0.7100 ▼
segment0	0.9757	0.9306 ▼	**0.9757**	0.8900 ▼
shuttle-c0-vs-c4	0.9946	**0.9946**	**0.9946**	0.9400 ▼
vehicle2	0.9825	**0.9825**	0.9525 ▼	0.9000 ▼
vowel0	1.0000	0.9966	0.9757 ▼	0.9400 ▼
winequality-red-4	0.6742	0.6599	0.6255 ▼	0.5500 ▼
yeast1	0.7430	0.7403	0.7189 ▼	0.6500 ▼

[8] O. Loyola-González, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and M. García-Borroto, “Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases,” *Neurocomputing*, vol. 175, pp. 935–947, 2016.

[9] D. L. Wilson, “Asymptotic properties of nearest neighbor rules using edited data,” *Systems, Man and Cybernetics, IEEE Transactions on*, no. 3, pp. 408–421, 1972.

[10] I. Mani and I. Zhang, “knn approach to unbalanced data distributions: a case study involving information extraction,” in *Proceedings of Workshop on Learning from Imbalanced Datasets*, 2003.

[11] J. Laurikkala, “Improving identification of difficult small classes by balancing class distribution,” in *Artificial Intelligence in Medicine*, ser. Lecture Notes in Computer Science, S. Quaglini, P. Barahona, and S. Andreassen, Eds. Springer Berlin Heidelberg, 2001, vol. 2101, pp. 63–66. [Online]. Available: http://dx.doi.org/10.1007/3-540-48229-6_9

[12] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis, “kernlab – an S4 package for kernel methods in R,” *Journal of Statistical Software*, vol. 11, no. 9, pp. 1–20, 2004. [Online]. Available: http://www.jstatsoft.org/v11/i09/

[13] P. Kuba, P. Brazdil, C. Soares, A. Woznica *et al.*, “Exploiting sampling and meta-learning for parameter setting for support vector machines,” 2002.

[14] “The promise repository of empirical software engineering data,” 2015.

[15] J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, “Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework,” *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, no. 255-287, p. 11, 2010.

[16] M. Lichman, “UCI machine learning repository,” 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[17] M. Shepperd, Q. Song, Z. Sun, and C. Mair, “Data quality: Some comments on the nasa software defect datasets,” *IEEE Trans. Softw. Eng.*, vol. 39, no. 9, pp. 1208–1215, Sep. 2013. [Online]. Available: http://dx.doi.org/10.1109/TSE.2013.11

[18] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2016. [Online]. Available: https://www.R-project.org/

[19] M. K. C. from Jed Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, the R Core Team, M. Benesty, R. Lescarbeau, A. Ziem, L. Scrucca, Y. Tang, and C. Candan., *caret: Classification and Regression Training*, 2016, r package version 6.0-68. [Online]. Available: https://CRAN.R-project.org/package=caret

[20] L. Gautier, “rpy2: A simple and efficient access to r from python,” *URL http://rpy. sourceforge. net/rpy2. html*, 2008.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, “Scikit-learn: Machine learning in python,” *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.