Journal of Intelligent Learning Systems and Applications, 2015, 7, 104-116
Published Online November 2015 in SciRes. http://www.scirp.org/journal/jilsa
http://dx.doi.org/10.4236/jilsa.2015.74010

A KNN Undersampling Approach for Data Balancing

Marcelo Beckmann, Nelson F. F. Ebecken, Beatriz S. L. Pires de Lima

Civil Engineering Program/COPPE, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
Email: beckmann.marcelo@gmail.com

Received 30 April 2015; accepted 8 November 2015; published 11 November 2015

Abstract

In supervised learning, the imbalanced number of instances among the classes in a dataset can make the algorithms to classify one instance from the minority class as one from the majority class. With the aim to solve this problem, the KNN algorithm provides a basis to other balancing methods. These balancing methods are revisited in this work, and a new and simple approach of KNN undersampling is proposed. The experiments demonstrated that the KNN undersampling method outperformed other sampling methods. The proposed method also outperformed the results of other studies, and indicates that the simplicity of KNN can be used as a base for efficient algorithms in machine learning and knowledge discovery.

Keywords

Machine Learning, Class Overlaping, Imbalanced Datases

1. Introduction

When dealing with supervised learning, one of the main problems in classification activities lies in the treatment of datasets where one or more classes have a minority quantity of instances. This condition denotes an imbalanced dataset, which makes the algorithm to incorrectly classify one instance from the minority class as belonging to the majority class, and in highly skewed datasets, this is also denoted as a "needle in the haystack" problem [1], due to the high number of instances from a class overcoming one or more minority classes. Nevertheless, in most of cases the minority class represents an abnormal event in a dataset, and usually this is the most interesting and valuable information to be discovered.

Learning from imbalanced datasets is still considered an open problem in data mining and knowledge discovery, and needs real attention from the scientific community [2]. The experiments performed in [3] demonstrated the class overlapping is commonly associated with the class imbalance problem. An empirical study was per-

How to cite this paper: Beckmann, M., Ebecken, N.F.F. and de Lima, B.S.L.P. (2015) A KNN Undersampling Approach for Data Balancing. *Journal of Intelligent Learning Systems and Applications*, **7**, 104-116.
http://dx.doi.org/10.4236/jilsa.2015.74010

formed by [1], in order to identify causes of why classifiers perform worse in the presence of class imbalance. In [4] the authors conducted a taxonomy of methods applied to correct or mitigate this problem, and in this study three main approaches were found: data adjusting, cost sensitive learning, and algorithm adjusting. In the data adjusting, there are two main sub-approaches: creation of instances from minority class (oversampling), and removal of instances from majority class.

This work is focused in the data adjusting algorithms, and a proposal of a KNN undersampling (KNN-Und) algorithm will be presented. The KNN-Und is a very simple algorithm, and basically it uses the neighbor count to remove instances from majority class. Despite its simplicity, the classification experiments performed with KNN-Und balancing resulted in better performance of G-Mean [5] and AUC [6], in most of the 33 datasets, if compared with three methods based on KNN: SMOTE [7], ENN [8], NCL [9], and the random undersampling method. The KNN-Und balancing also performed better than the results published previously by [10] in 11 of 15 datasets, and had higher average values of G-Mean and AUC, than the evolutionary algorithm proposed by [11]. The results obtained in the experiments show that KNN-Und and other balancing methods based on KNN are an interesting approach to solve the imbalanced dataset problem. Instead of generating new synthetic data as oversampling methods, especially when the datasets are approaching petabytes of size [12], the oriented removal of majority instances can be a better solution than to create more data.

This paper is organized as follows. In Session 2 a literature review about KNN balancing methods is presented, in Session 3 the KNN-Und methodology is explained in more details. In section 4 the experiments conducted in this work will be presented, compared and commented, followed by the conclusions.

Imbalanced Dataset Definition

This section establishes some notations that will be used in this work.

Given the training set T with m examples and n attributes, where $T = \{x_i, y_i\}, i = 1, \cdots, m$, and where $x_i \in X$ is an instance in the set of attributes $X = \{a_1, \cdots, a_n\}$, and $y_i \in Y$ is an instance in the set of classes $Y = \{1, \cdots, c\}$, there is a subset with positive instances $P \subset X$, and a subset of negative instances $N \subset X$, where $|P| < |N|$. All subset of P created by sampling methods will be denominated S. The pre-processing strategies applied to datasets aims to balance the training set T, such as $|P| \cong |N|$.

2. Literature Review

Along the years, a great effort was done in the scientific community in order to solve or mitigate the imbalanced dataset problem. Specifically for KNN, there are several balancing methods based on this algorithm. This section will provide a bibliographic review about the KNN and its derivate algorithms for dataset balancing. Also, the random oversampling and undersampling methods, the class overlapping problem, and evaluation measures will be reviewed.

2.1. KNN Classifier

The k Nearest Neighbor (KNN) is a supervised classifier algorithm, and despite his simplicity, it is considered one of the top 10 data mining algorithms [13]. It creates a decision surface that adapts to the shape of the data distribution, enabling them to obtain good accuracy rates when the training set is large or representative. The KNN was introduced initially by [14], and it was developed with the need of perform discriminant analysis when reliable parametric estimates of probability densities are unknown or difficult to determine.

The KNN is a nonparametric lazy learning algorithm. It is nonparametric because it does not make any assumptions on the underlying data distribution. Most of the practical data in the real world does not obey the typical theoretical assumptions made (for example, Gaussian mixtures, linear separability, etc....). Nonparametric algorithms like KNN are more suitable on these cases [15] [16].

It is also considered a lazy algorithm. A lazy algorithm works with a nonexistent or minimal training phase but a costly testing phase. For KNN this means the training phase is fast, but all the training data is needed during the testing phase, or at the least, a subset with the most representative data must be present. This contrasts with other techniques like SVM, where you can discard all nonsupport vectors.

The classification algorithm is performed according to the following steps:

1. Calculate the distance (usually Euclidean) between a x_i instance and all instances of the training set T;
2. Select the k nearest neighbors;
3. The x_i instance is classified (labeled) with the most frequent class among the k nearest neighbors. It is also possible to use the neighbors' distance to weight the classification decision.

The value of k is training-data dependent. A small value of k means that noise will have a higher influence on the result. A large value makes it computationally expensive and defeats the basic philosophy behind KNN: points that are close might have similar densities or classes. Typically in the literature are found odd values for k, normally with $k = 5$ or $k = 7$, and [15] reports $k = 3$ allowing to obtain a performance very close to the Bayesian classifier in large datasets. An approach to determine k as a function (1) of data size m is proposed in [16].

$$k = odd\left(\sqrt{m}\right) \tag{1}$$

The algorithm may use other distance metrics than Euclidean [17] [18].

2.2. SMOTE

The SMOTE algorithm proposed by [7] is one of the most known oversampling techniques, being successful in several areas of application, being also a base for other oversampling algorithms [9] [19]-[22].

The SMOTE executes the balancing of a P set of minority instances, creating n synthetic instances from each instance x_i of the P set. The synthetic instance is created based in a minority instance and its nearest neighbors. One synthetic instance is generated based in the instances x_i and $\hat{x}_i$, being $\hat{x}_i$ an instance randomly selected among the k nearest neighbors (KNN) of x_i, and δ as a random number between 0 and 1, according the Equation (2). The process is repeated n times for each instance x_i from the P set, where $n = b/100$, and b is a parameter that defines the percentage of oversampling required to balance the dataset.

$$x_{\text{synthetic}} = x_i + \left(\hat{x}_i - x_i\right) \cdot \delta \tag{2}$$

Figure 1 shows the SMOTE process with $k = 5$. Starting from (a), there is an imbalanced dataset, where (−) belongs to majority instances, also known as negative instances, and (+) belongs to minority instances, also known as positive instances. In (b) The KNN selects 5 nearest neighbors from a minority instance $\hat{x}_i$. In (c) one of the five nearest neighbors $\hat{x}_i$ is randomly selected. In (d) a new synthetic instance is generated with random attributes between x_i and $\hat{x}_i$. The process is repeated for every minority instance (+) from the subset P.

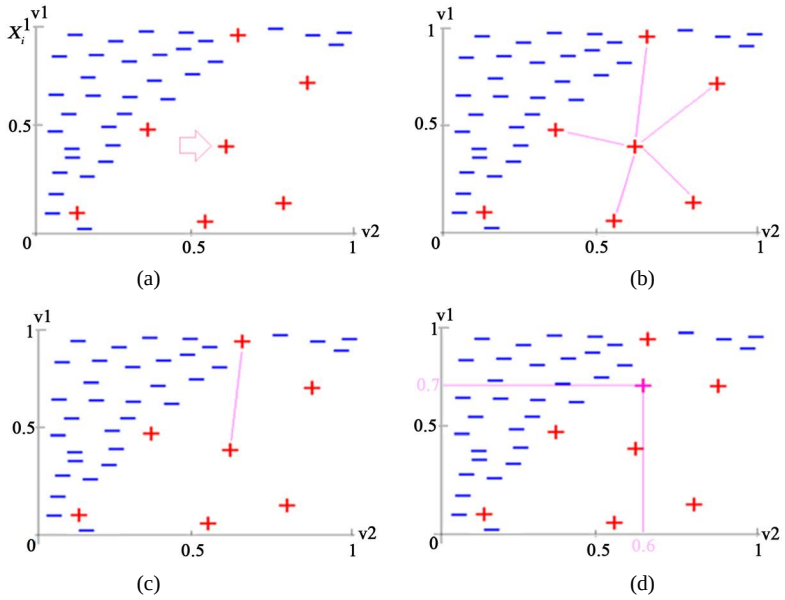

Figure 1. SMOTE process for $k = 5$. (a) An imbalanced dataset, with negative (−) and positive (+) instances. An instance x_i is selected; (b) the $k = 5$ nearest instances (neighbors) of x_i are selected; (c) one of the $k = 5$ neighbors $\hat{x}_i$, is randomly selected; (d) a new synthetic instance is created with the random values of v1 and v2 between x_i and $\hat{x}_i$.

An extensive comparison of several oversampling and undersampling methods was performed in [23]. The authors concluded the SMOTE combined with Tomek Links [24] and ENN methods [8] presented better performance in 50% of the experiments. Nevertheless, the SMOTE algorithm alone had better performance in 16% of the cases, and in most of the cases, it presented similar results in terms of AUC, if compared with the combined methods.

According the experiments conducted in [25], one of the weaknesses of *SMOTE* lies in the fact all the positive instances acts as a base for synthetic instance generation. The authors argue such strategy doesn't take into account that not always a homogeneous distribution of synthetic instances is applicable to an unbalancing problem; as such practice could cause overfitting and class overlapping. Another weakness reported is the result variance, caused by the random characteristics existing in some points of the algorithm.

2.3. Edited Nearest Neighbor Rule (ENN)

The ENN method proposed by [8], removes the instances of the majority class whose prediction made by KNN method is different from the majority class. So, if an instance $x_i \in N$ has more neighbors of a different class, this instance x_i will be removed. The ENN works according to the steps below:

1. Obtain the k nearest neighbors of x_i, $x_i \in N$;
2. x_i will be removed if the number of neighbors from another class is predominant;
3. The process is repeated for every majority instance of the subset N.

According to the experiments conducted in [26], the ENN method removes both the noisy examples as borderline examples, providing a smoother decision surface.

2.4. Neighbor Cleaning Rule (NCL)

The Neighbor Cleaning Rule (NCL) proposed by [9], consists in improving the ENN method for two-classes problems in the following way: for each example $x_i \in T$, find its $k = 3$ nearest neighbors. If x_i belongs to the majority class and there is a prediction error related to its nearest neighbors, x_i will be removed. If x_i belongs to the minority class and there is a prediction error related to its nearest neighbors, the nearest neighbors belonging to the majority class will be removed.

2.5. Random Sampling

This is one of the simplest strategies for data sets adjusting, and basically consists in the random removal (undersampling) and addition (oversampling) of instances. For oversampling, instances of the positive set P are randomly selected, duplicated and added to the set T. For undersampling, the instances from negative set N are randomly selected for removal.

Although both strategies have the similar operation and brings some benefit than simply classifying without any preprocessing [1] [9], they also introduce problems in learning. For the instances removal, there is the risk of removing important concepts related to the negative class. In the case of adding positive instances, the risk is to create over adjustment (overfitting), *i.e.*, a classifier can construct rules that apparently are precise, but in fact cover only a replicated example.

2.6. Class Overlapping Problem

According the experiments of [3], the low classification performance on imbalanced datasets is not associated only to the class distribution, but is also related to class overlapping. The authors concluded that normally in highly skewed datasets, the problem of "needle in the haystack" comes together with a class overlapping problems.

2.7. Evaluation Measures

In supervised learning, it is necessary to use some measure to evaluate the results obtained with a classifier algorithm. The confusion matrix from **Table 1**, also known as contingency table, is frequently applied for such purposes, providing not only the count of errors and hits, but also the necessary variables to calculate other measures.

The confusion matrix is able to represent either two class or multiclass problems. Nevertheless, the research and literature related to imbalanced datasets is concentrated in two class problems, also known as binary or binomial

Table 1. Confusion matrix

	Positive prediction	Negative prediction
Positive class	True Positive (TP)	False Negative (FN)
Negative class	False Positive (FN)	True Negative (TN)

problems, which the less frequent class is named as positive, and the remaining classes are merged and named as negative.

Some of the most known measures derived from this matrix are the error rate (3) and the accuracy (4). Nevertheless, such measures are not appropriated to evaluate imbalanced datasets, because they do not take into account the number of examples distributed among the classes. On the other hand, there are measures that compensate this disproportion in their calculation. The Precision (5), Recall (8) and F-Measure [27] are appropriated when the positive class is the main concern. The G-Mean, ROC and AUC are appropriated when the performance of both classes (positive and negative) are important.

$$\text{Error} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}} \tag{3}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}} \tag{4}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

In this work, both classes are considered as equal importance, therefore, the measures G-Mean and AUC will be used to evaluate the experiments.

The G-Mean [5] verifies the performance in both classes, taking into account the distribution between them, by computing the geometric average between the true positives and true negatives (6).

$$G\text{-Mean} = \sqrt{\text{TP} * \text{TN}} \tag{6}$$

The Receiver Operating Characteristics (ROC) chart, also denominated ROC Curve, has been applied in detection, signal analysis since the Second World War, and recently in data mining and classification. It consists of a two dimensions chart, were the y-axis refers to Sensitivity or Recall (7), and the x-axis calculated as 1-Especificity (8). According to [6] there are several points in this chart that deserve attention. By analyzing this chart it is possible to identify not only the classifier performance, but also to deduce some classifier behaviors like: conservative, aggressive, or aleatory.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7}$$

$$\text{Especificity} = \frac{\text{TN}}{\text{FP} + \text{TN}} \tag{8}$$

The AUC measure (9) synthetizes as a simple scalar the information represented by a ROC chart, and is insensitive to class imbalance problems.

$$\text{AUC} = \Phi\left(\frac{\delta}{\sqrt{\phi_{pos} + \phi_{neg}}}\right) \tag{9}$$

where Φ is the normal cumulative distribution, δ is the Euclidean distance between the class centroids of two classes, and ϕ_{pos}, and ϕ_{neg} are the standard deviation from the positive and negative classes. An algorithm to calculate AUC is also provided in [6].

3. Proposed Method: KNN-Und

The KNN-Und method works removing instances from the majority classes based on his *k* nearest neighbors,

and works according to the steps below:

1. Obtain the k nearest neighbors for $x_i \in N$;
2. x_i will be removed if the count of its neighbor is greater or equal to t;
3. The process is repeated for every majority instance of the subset N.

The parameter t defines the minimum count of neighbors around x_i belonging to the P (minority) subset. If this count is greater or equal t, the instance x_i will be removed from the training set T. The valid values of $t$are $1 \le t \le k$ and as lower t is, as aggressive is the undersampling. This algorithm can also be used in multiclass problems, as in the negative subset N can contain instances from several majority classes. The KNN-Und algorithm was developed as a preprocessing plug in in Weka platform [28].

If compared with ENN, the KNN-Und has a more aggressive behavior in terms of instance removal, because KNN-Und does not depend of a wrong prediction of KNN to remove an instance $x_i \in N$. KNN-Und only acts in the class overlapping areas, because an instance from majority class only will be removed if a number t of instances from other classes are present in its neighborhood. In the cases that an instance of the majority class is not surrounded by t instances of other classes, that instance will not be removed. This situation only occurs in non-overlapping areas. Despite this behavior, in our experiments $t = 1$ was kept in most of the cases, because the KNN-Und only acts in overlapping areas. The nonoverlapping areas, which are far from the decision surface are kept untouchable. This explains why the KNN-Und can also be used to solve the class-overlapping problem, which is commonly associated with imbalanced datasets [3]. Nevertheless, in highly skewed problems the KNN-Und is not efficient to balance the dataset. In these cases, the combination of KNN-Und with another sampling method could improve the results.

The KNN-Und can be considered a very simple algorithm, and has the advantage to be a deterministic method, since different of other methods, there is no random component. In the literature review, only one study [29] has mentioned a similar methodology and application so far, but the results published previously demonstrated this alternative was not very well exploited at that time.

4. Experiments

In this section, the experiments to validate the applicability of KNN-Und are conducted. The **Table 2** depicts the 33 datasets prepared for the experiments, ordered by imbalance rate (IR) [30], which is the rate between the quantity of negative and positive instances. All the 33datasets are originated from UCI machine learning repository [31]. For the data sets with more than two classes, one or more classes with fewer examples were selected as the positive class, and collapsed the remainder as the negative class. The datasets were balanced with KNN-Und and submitted to a classifier, then the evaluation measures were compared with three methods based on KNN: SMOTE [7], ENN [8], NCL [9], and the random undersampling method. The performance of KNN-Und was also compared with the published results of two other studies [10] [11]. All the algorithms tested in this work were implemented in Weka platform [28].

In all datasets and algorithms that uses KNN, the parameter k was determined according to (1). The parameter t was adjusted in order to control the undersampling level with KNN-Und method, and in most of datasets with IR < 2, this parameter was set to $t > 1$ to control the excessive undersampling. The **Table 2** shows the values of k and t for each dataset, and the respective under sampling effect of KNN-Und on the negative (majority) class and resulting IR. The parameters for SMOTE and Random Undersampling methods were adjusted to obtain a balanced class distribution. The ENN and NCL methods do not have a balancing control parameter.

The classification results in terms of AUC and G-Mean are presented in **Table 3** and **Table 4**, respectively.

Those tables show the results averaged over 10 runs and the standard deviation between parentheses for the 33 datasets.

The classification results in terms of AUC with KNN-Und data preparation were compared with our previous work [10]. This paper proposed a Genetic Algorithm (GA) as an oversampling method, with the aim to evolve sub regions filled with synthetic instances to adjust imbalanced datasets. To reproduce here the same experiments conditions as before, the classifier used is the C4.5 decision tree algorithm [32], with 25% of pruning and 10-fold cross validation. The Euclidean distance was used for numeric attributes, and the superposition distance for nominal attributes [17] [18].

The same experiment setup was applied forC4.5 classifier without balancing (as a baseline comparison) and for the others balancing methods: SMOTE, ENN, NCL and Random Undersampling. The last columns of **Table 3** and **Table 4** presents the results of the proposed balancing method, KNN-Und, with C4.5 and 1-NN classifier.

Table 2. Datasets ordered by IR before balancing with KNN-Und and the respective effect after KNN-Und.

Dataset	# Examples	Classes (min, maj)	# Classes before (min/maj)	IR before	*k*	*t*	# Classes after (min, maj)	IR After
GlassBWNFP	214	(build-wind-non-float, remainder)	(76, 138)	1.82	15	5	(76, 76)	1.00
EcoliCP-IM	220	(im, remainder)	(77, 143)	1.86	15	1	(77, 122)	1.58
Pima	768	(1,0)	(268, 500)	1.87	27	5	(268, 228)	0.85
GlassBWFP	214	(build-wind-float, remainder)	(70, 144)	2.06	15	3	(70, 67)	0.96
German	1000	(bad, good)	(300, 700)	2.33	31	7	(300, 326)	1.09
Post-Oper	90	(S, remainder)	(24, 66)	2.75	9	4	(24, 15)	0.63
Habermann	306	(die, survive)	(81, 225)	2.77	17	4	(81, 104)	1.28
Splice-ie	3176	(ie, remainder)	(768, 2422)	3.15	57	9	(768, 773)	1.01
Splice-ei	3176	(ei, remainder)	(767, 2423)	3.15	57	9	(767, 572)	0.75
GlassNW	214	(non-wind-glass, remainder)	(51, 163)	3.19	15	1	(51, 141)	2.76
VehicleVan	846	(van, remainder)	(199, 647)	3.25	29	1	(199, 394)	1.98
EcoliIM	336	(im, remainder)	(77, 259)	3.36	19	1	(77, 186)	2.42
Letter-vowel	20000	(all vowels, remainder)	(3878, 16122)	4.15	161	1	(3878, 4543)	1.17
New-Thyroid	215	(hypo, remainder)	(30, 185)	6.16	15	1	(30, 172)	5.73
Segment1	2310	(brickface, sky)	(330, 1980)	6.00	49	1	(330, 1695)	5.14
EcoliIMU	336	(iMU, remainder)	(35, 301)	8.60	19	1	(35, 236)	6.74
Optdigits0	5564	(0, remainder)	(554, 5066)	9.14	75	1	(554, 4951)	8.94
Satimage4	6435	(4, remainder)	(626, 5809)	9.28	91	1	(626, 4880)	7.80
Vowel0	990	(0, remainder)	(90, 900)	10.00	31	1	(90, 881)	8.79
Flag	194	(white, remainder)	(17, 177)	10.41	15	1	(17, 61)	3.59
GlassVWFP	214	(ve-win-float-proc, remainder)	(17, 197)	11.58	15	1	(17, 151)	8.88
EcoliOM	336	(om, remainder)	(20, 316)	15.80	19	1	(20, 279)	13.95
GlassContainers	214	(containers, remainder)	(13, 201)	15.46	15	1	(13, 175)	13.46
Abalone9-18	731	(18, 9)	(42, 689)	16.40	81	1	(42, 46)	1.10
GlassTableware	214	(tableware, remainder)	(9, 205)	22.77	15	1	(9, 175)	19.44
YeastCyt-Pox	483	(Pox, Cyt)	(20, 463)	23.15	21	1	(20, 344)	17.20
YeastME2	1484	(ME2, remainder)	(51, 1433)	28.10	39	1	(51, 1015)	19.90
YeastME1	1484	(ME1, remainder)	(44, 1440)	32.72	39	1	(44, 1313)	29.84
YeastEXC	1484	(EXC, remainder)	(35, 1449)	41.40	39	1	(35, 1129)	32.26
Car	1728	(good, remainder)	(69, 1659)	24.04	41	1	(69, 556)	8.06
Letter-A	20000	(a, remainder)	(789, 19211)	24.34	141	1	(789, 16680)	21.14
Nursery	12960	(rec + very − rec, remainder)	(330, 12630)	38.27	113	1	(330, 6692)	20.28
Abalone19	4177	(19, remainder)	(32, 4145)	129.53	65	1	(32, 2983)	93.21

Table 3. Classification results—AUC.

Dataset	Best result in [10]	C4.5	SMOTE + C4.5	ENN + C4.5	NCL C4.5	Random Undersampling + C4.5	KNN-Und + C4.5	KNN-Und + 1-NN
GlassBWNFP	-	74.72 (0)	72.81 (0.56)	77.86 (0)	78.33 (0)	71.97 (2.69)	85.14 (0)	**93.46 (0)**
EcoliCP-IM	-	97.32 (0)	97.33 (0)	97.32 (0)	**99.23 (0)**	96.59 (0.34)	98.48 (0)	98.24 (0)
Pima	82.72 (0.48)	75.14 (0)	73.10 (0.33)	79.71 (0)	91.16 (0)	73.39 (2.06)	92.22 (0)	**92.23 (0)**
GlassBWFP	-	82.10 (0)	83.27 (2.85)	85.09 (0)	96.16 (0)	86.58 (1.49)	97.86 (0)	**98.96 (0)**
German	76.99 (0.82)	63.90 (0)	64.40 (1.29)	67.83 (0)	57.37 (0)	65.53 (1.36)	**84.44 (0)**	84.20 (0)
Post-Oper	69.68 (1.69)	42.80 (0)	65.30 (3.42)	42.80 (0)	4.17 (0)	37.22 (7.48)	**71.39 (0)**	-
Habermann	73.73 (0.74)	56.4 (0)	58.81 (2.32)	56.11 (0)	44.44 (0)	61.05 (1.00)	**85.49 (0)**	81.22 (0)
Splice-ie	**98.84 (0.20)**	96.02 (0)	96.54 (0.39)	96.63 (0)	97.61 (0)	95.80 (0.46)	97.73 (0)	97.96 (0)
Splice-ei	96.65 (0.22)	96.70 (0)	97.74 (0.16)	97.35 (0)	**99.07 (0)**	97.24 (0.49)	98.51 (0)	96.46 (0)
GlassNW	-	87.10 (0)	96.54 (2.94)	91.59 (0)	97.38 (0)	91.61 (1.97)	95.08 (0)	**97.45 (0)**
VehicleVan	99.01 (0.32)	93.43 (0)	93.42 (0.67)	94.99 (0)	97.39 (0)	95.34 (1.89)	99.14 (0)	**100 (0)**
EcoliIM	-	91.82 (0)	88.94 (0.16)	92.02 (0)	94.27 (0)	91.08 (0.12)	**98.71 (0)**	98.10 (0)
Letter-vowel	95.38 (0.22)	94.98 (0)	94.55 (0.22)	95.02 (0)	98.11 (0)	93.54 (0.38)	**99.11 (0)**	-
New-Thyroid	**99.81 (0.23)**	93.98 (0)	93.88 (2.17)	93.98 (0)	96.67 (0)	94.83 (1.32)	95.63 (0)	95.97 (0)
Segment1	-	98.30 (0)	97.84 (0)	97.70 (0)	98.67 (0)	98.87 (0.36)	99.24 (0)	**100 (0)**
EcoliIMU	97.19 (0.24)	80.09 (0)	85.86 (1.60)	84.15 (0)	95.22 (0)	91.45 (0.98)	97.14 (0)	**98.43 (0)**
Optdigits0	-	96.53 (0)	98.54 (0.27)	96.75 (0)	96.26 (0)	98.21 (0.54)	97.98 (0)	**99.96 (0)**
Satimage4	84.14 (0.52)	83.27 (0)	83.25 (1.07)	87.10 (0)	95.04 (0)	87.25 (0.59)	96.56 (0)	**99.27 (0)**
Vowel0	-	97.96 (0)	95.76 (1.92)	97.96 (0)	98.86 (0)	98.64 (0.87)	99.09 (0)	**100 (0)**
Flag	**86.03 (1.86)**	43.17 (0)	42.25 (5.03)	43.17 (0)	76.98 (0)	73.08 (10.55)	84.14 (0)	-
GlassVWFP	96.81 (1.62)	96.55 (0)	96.13 (0.16)	99.48 (0)	**99.69 (0)**	95.07 (2.75)	99.67 (0)	97.64 (0)
EcoliOM	-	79.63 (0)	79.27 (2.85)	79.63 (0)	84.61 (0)	91.04 (4.19)	85.15 (0)	**100 (0)**
GlassContainers	-	79.54 (0)	81.30 (3.56)	76.21 (0)	90.94 (0)	91.49 (2.99)	**95.31 (0)**	79.04 (0)
Abalone9-18	-	65.14 (0)	63.92 (4.32)	65.14 (0)	87.36 (0)	62.31 (2.67)	92.34 (0)	**99.02 (0)**
GlassTableware	-	99.54 (0)	94.63 (3.98)	99.54 (0)	**100 (0)**	**100 (0)**	99.87 (0)	**100 (0)**
YeastCyt-Pox	-	48.46 (0)	63.09 (0)	48.46 (0)	63.38 (0)	76.95 (9.04)	63.86 (0)	**87.66 (0)**
YeastME2	-	74.12 (0)	75.60 (0.85)	74.12 (0)	92.73 (0)	79.18 (2.78)	**95.87 (0)**	91.09 (0)
YeastME1	-	89.42 (0)	83.22 (0)	91.46 (0)	98.01 (0)	95.38 (0.44)	96.22 (0)	**100 (0)**
YeastEXC	-	80.54 (0)	81.08 (1.50)	80.54 (0)	80.43 (0)	78.23 (4.44)	89.70 (0)	**91.77 (0)**
Car	-	49.32 (0)	99.25 (0.03)	49.32 (0)	98.93 (0)	92.16 (1.44)	99.97 (0)	**100 (0)**
Letter-A	99.95 (0.09)	98.54 (0)	98.05 (0.38)	98.64 (0)	**99.97 (0)**	97.66 (0.55)	99.00 (0)	-
Nursery	99.38 (0.30)	98.90 (0)	99.86 (0)	98.90 (0)	**100 (0)**	98.81 (0.48)	**100 (0)**	-
Abalone19	-	47.47 (0)	43.08 (2.70)	47.47 (0)	51.22 (0)	64.95 (2.32)	**71.06 (0)**	67.17 (0)

Table 4. Classification results—G-mean.

Dataset	C4.5	SMOTE + C4.5	ENN + C4.5	NCL + C4.5	Random Undersampling + C4.5	KNN-Und + C4.5	KNN-Und + 1-NN
GlassBWNFP	73.39 (0)	73.04 (2.00)	75.44 (0)	76.01 (0)	74.05 (4.01)	85.27 (0)	**92.51 (0)**
EcoliCP-IM	98.34 (0)	**98.35 (0)**	98.34 (0)	98.19 (0)	97.52 (0.40)	97.47 (0)	98.03 (0)
Pima	69.71 (0)	70.95 (1.25)	75.05 (0)	87.48 (0)	73.40 (2.63)	89.70 (0)	**91.30 (0)**
GlassBWFP	80.84 (0)	79.99 (1.99)	83.87 (0)	92.48 (0)	84.79 (1.57)	96.36 (0)	**98.54 (0)**
German	57.43	61.26 (1.13)	63.14 (0)	61.60 (0)	65.74 (1.89)	**84.30 (0)**	83.59 (0)
Post-Oper	0 (0)	62.32 (2.30)	0 (0)	0 (0)	31.15 (13.30)	**77.46 (0)**	-
Habermann	44.89 (0)	54.33 (3.54)	47.24 (0)	0 (0)	62.39 (2.19)	**83.18 (0)**	79.83 (0)
Splice-ie	94.63 (0)	96.26 (0.31)	94.94 (0)	**98.33 (0)**	96.00 (0.34)	97.99 (0)	95.70 (0)
Splice-ei	96.92 (0)	97.42 (0.09)	97.03 (0)	**98.93 (0)**	96.81 (0.49)	98.84 (0)	93.78 (0)
GlassNW	89.63 (0)	96.28 (2.79)	93.33 (0)	96.96 (0)	89.95 (4.03)	**97.01 (0)**	**97.01 (0)**
VehicleVan	92.06 (0)	91.76 (0.61)	92.76 (0)	96.65 (0)	93.46 (1.75)	99.11 (0)	**100 (0)**
EcoliIM	83.89 (0)	81.99 (0.07)	91.56 (0)	95.37 (0)	91.60 (0.08)	97.50 (0)	**98.03 (0)**
Letter-vowel	92.74 (0)	92.02 (0.15)	92.97 (0)	97.71 (0)	91.88 (0.32)	**98.96 (0)**	-
New-Thyroid	93.84 (0)	93.34 (0.68)	93.84 (0)	**96.61 (0)**	93.82 (1.93)	94.32 (0)	**96.61 (0)**
Segment1	98.30 (0)	97.92 (0)	98.22 (0)	98.19 (0)	98.83 (0.34)	99.36 (0)	**100 (0)**
EcoliIMU	74.45 (0)	69.44 (5.58)	83.79 (0)	93.70 (0)	94.65 (1.83)	98.35 (0)	**98.56 (0)**
Optdigits0	97.81 (0)	98.67 (0.17)	97.88 (0)	97.74 (0)	98.51 (0.37)	98.34 (0)	**99.91 (0)**
Satimage4	79.01 (0)	81.65 (1.34)	83.22 (0)	94.82 (0)	87.95 (0.54)	95.82 (0)	**99.28 (0)**
Vowel0	97.48 (0)	93.55 (1.41)	97.48 (0)	98.17 (0)	98.54 (1.01)	96.94 (0)	**100 (0)**
Flag	0 (0)	34.97 (3.59)	0 (0)	51.82 (0)	66.37 (9.44)	**79.11 (0)**	-
GlassVWFP	96.52 (0)	96.22 (0.11)	99.48 (0)	**99.69 (0)**	92.88 (2.46)	99.67 (0)	97.01 (0)
EcoliOM	82.87 (0)	79.70 (3.16)	82.87 (0)	83.20 (0)	88.91 (3.26)	85.82 (0)	**100 (0)**
GlassContainers	73.01 (0)	79.87 (2.78)	77.26 (0)	86.13 (0)	89.57 (2.48)	**95.26 (0)**	78.45 (0)
Abalone9-18	48.33 (0)	39.11 (5.56)	48.33 (0)	86.88 (0)	61.54 (4.38)	91.80 (0)	**98.80 (0)**
GlassTableware	87.98 (0)	84.58 (7.60)	87.98 (0)	**100 (00)**	**100 (0)**	99.43 (0)	**100 (0)**
YeastCyt-Pox	0 (0)	74.08 (0)	0 (0)	49.82 (0)	74.48 (5.15)	44.72 (0)	**83.67 (0)**
YeastME2	55.66 (0)	50.22 (0)	55.66 (0)	84.81 (0)	76.72 (1.85)	84.92 (0)	**92.88 (0)**
YeastME1	86.27 (0)	82.34 (0)	89.00 (0)	95.27 (0)	96.22 (0.89)	97.66 (0)	**100.00 (0)**
YeastEXC	73.35 (0)	68.52 (1.09)	73.35 (0)	79.11 (0)	81.74 (2.49)	86.19 (0)	**89.44 (0)**
Car	0 (0)	98.61 (0.01)	0 (0)	98.33 (0)	88.16 (2.49)	99.82 (0)	**100 (0)**
Letter-A	97/03 (0)	96.79 (0.23)	99.64 (0)	**99.82 (0)**	97.30 (0.20)	98.69 (0)	-
Nursery	84.03 (0)	99.69 (0)	84.03 (0)	**100 (0)**	98.98 (0.33)	**100 (0)**	-
Abalone19	0 (0)	3.53 (7.89)	0 (0)	0 (0)	63.57 (2.69)	35.32 (0)	61.24 (0)

This last classifier was included to make a comparison with the evolutionary algorithm EBUS-MS-GM developed in [11].

The best result for each dataset is marked in bold.

Analyzing the AUC results in **Table 3**, it can be observed the KNN-Und with C4.5 algorithm outperformed in 19 of 33 datasets and had one dataset with equal result, if compared with the results of four different sampling methods. If compared with our previous results published in [10], the KNN-Und outperformed in 11 of 15 datasets.

Figure 2 illustrates the results of AUC using the balancing methods with C4.5 classifier for the 33 datasets. It shows the AUC values of KNN-Und (in green) at the top, or nearby, in all datasets.

The results in terms of G-Mean (**Table 4**) shows that the KNN-Und outperformed in 20 of 33 datasets, and one dataset with equal result. Different of GA, SMOTE and Random Undersampling methods, the KNN-Und, C4.5, ENN, and NCL have a deterministic behavior, which lead to most stable results with standard deviations equals to 0. The second best results were obtained with NCL algorithm, but an excessive undersampling was observed in datasets with IR < 2, which led to G-Mean values of 0.

Table 5 summarizes the count of the best results of the balancing methods with C4.5 classifier in terms of AUC and G-Mean. The KNN-Und has the highest scores.

These results can be explained by the fact that KNN-Und acts removing instances from the majority classes and at the same time cleaning the decision surface, reducing the class overlapping. **Figure 3** and **Figure 4** show the scatter plot of datasets EcoliIMU, and Satimage4, before and after the balancing methods. The points in blue belong to the majority class, the points in red to the minority class. These plots show the behavior of the methods, as described previously. The SMOTE algorithm performs a homogeneous distribution of synthetic instances around each positive instance. The ENN removes negative instances around the positive instances, and KNN-Und performs a more aggressive removal of negative instances in the decision surface region.

G-Mean and AUC values were not published by dataset for the evolutionary algorithm EUB-MS-GM in [11], so another comparison was done with the available results, that is, the average and standard deviation of G-Mean and AUC for the 28 evaluated datasets. **Table 6** compares the average results of KNN-Und and EUB-MS-GM methods. The KNN-Und results are at least 13 points higher than the EUB-MS-GM. It is not reasonable to do a comparison of standard deviations here, as the 28 datasets have independent results in both cases. One explanation for the obtained high values would be the 1-NN classifier used for comparison, that uses a decision boundary similar to KNN-Und, but the results for KNN-Und with C4.5 decision tree also had higher average values, showing that KNN-Und can also improve the classification results with other algorithms.

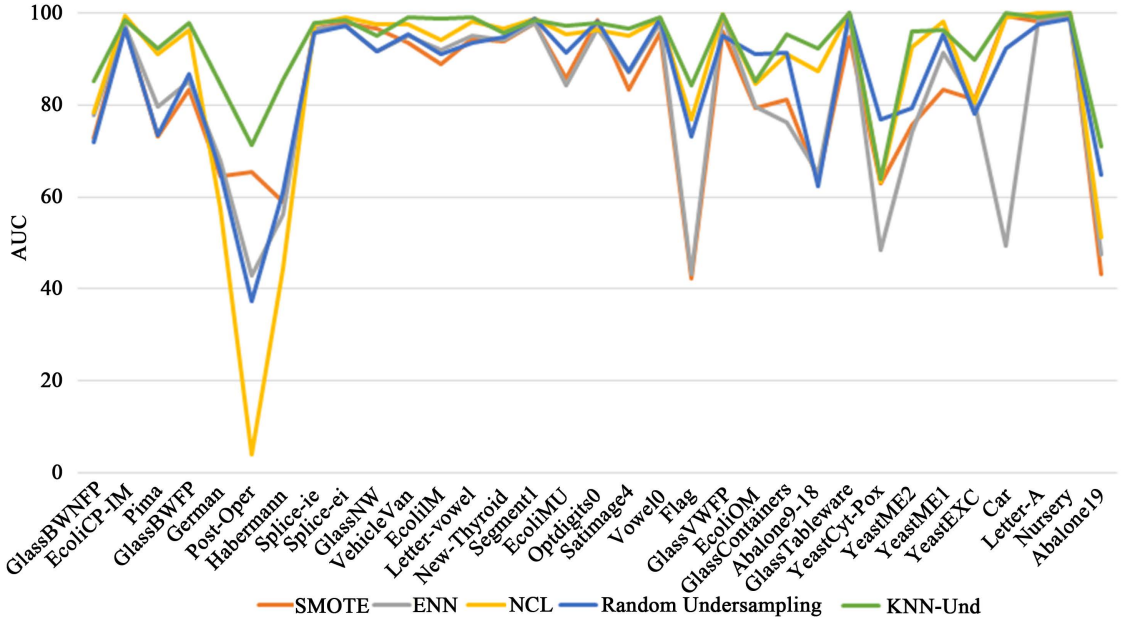

Figure 2. Comparison of classification with AUC after the balancing methods.

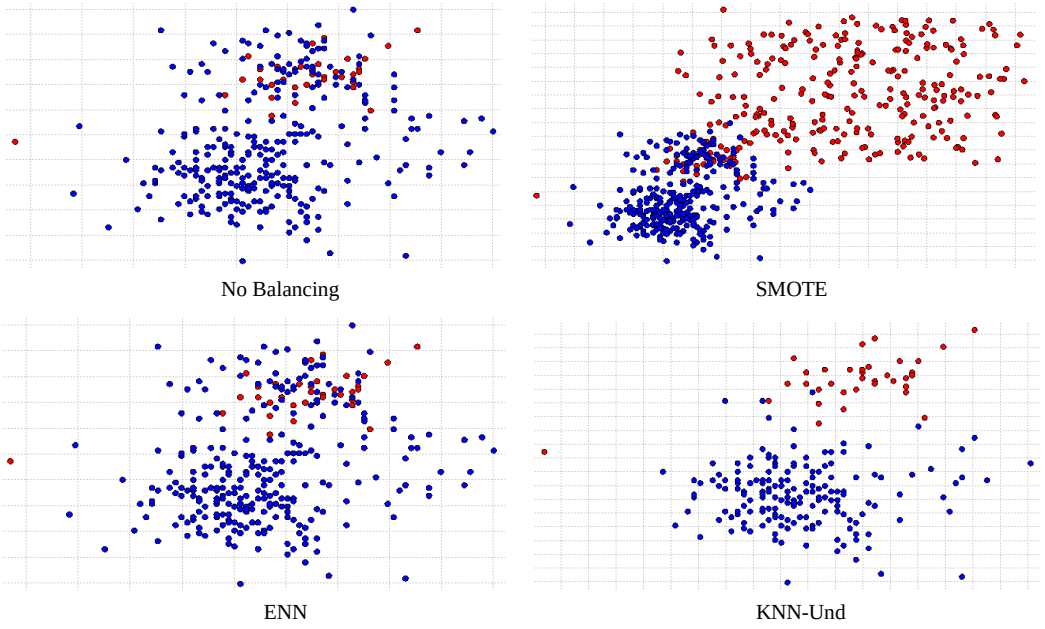

Figure 3. Scatter plot of EcoliIMU dataset before and after balancing methods. For x = aac (score of Amino acid content), y = alm1 (score of the ALOM membrane).

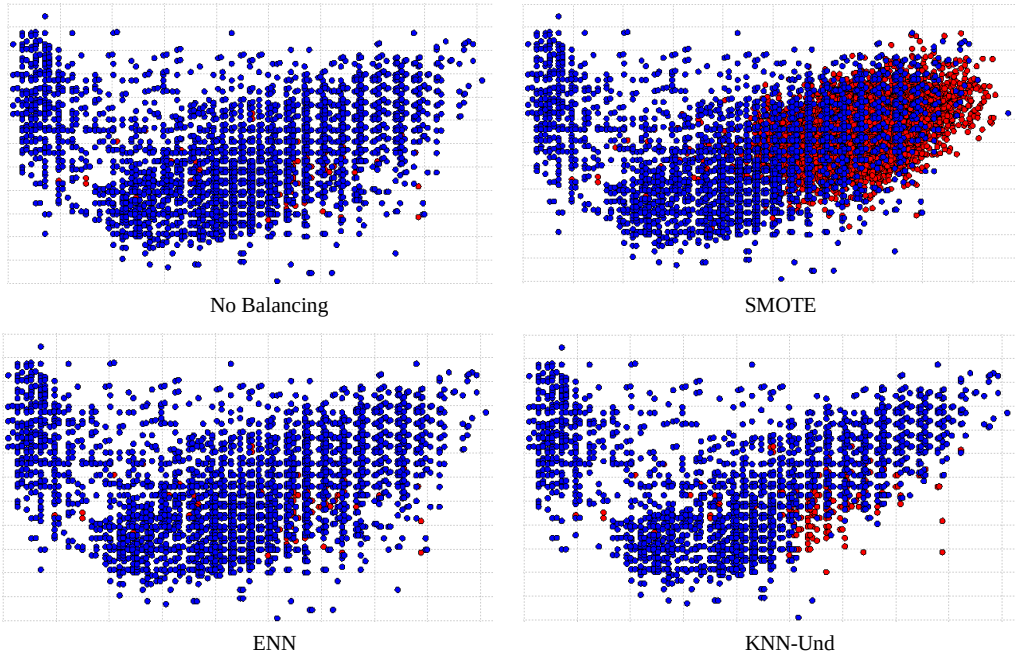

Figure 4. Scatter plot of Satimage4 dataset before and after balancing. For x = pixel column 6, y = pixel column 31.

Table 5. Summary of the classification best results in terms of AUC and G-Mean between KNN-Und and other balancing methods using C4.5.

Measure	Other methods	Equal results	KNN-Und
AUC	13	1	19
G-Mean	12	1	20

Table 6. Average of G-Mean and AUC from 28 the datasets used in [11].

Algorithm	1-NN		C4.5	
	G-Mean	AUC	G-Mean	AUC
KNN-Und	**93.72 (9.02)**	**94.47 (7.99)**	90.11 (15.52)	93.12 (8.77)
EBUS-MS-GM	79.71 (16.87)	80.85 (16.99)	-	-

5. Conclusions

This work presented a proposal of an algorithm, KNN-Und, to adjust datasets with imbalanced number of instances among the classes, also known as imbalanced datasets. The proposed method is an undersampling method, and is based on KNN algorithm, removing instances from the majority class based on the count of neighbors of different classes. The classification experiments conducted with the KNN Undersampling method on 33 datasets outperformed the results of other six methods, two studies based in evolutionary algorithms and the SMOTE, ENN, NCL and Random Undersampling methods.

The good results obtained with KNN Undersampling can be explained by the fact that KNN-Und acts removing instances from the majority classes, reducing this way the "needle in a haystack" effect, at the same time, cleaning the decision surface, reducing the class overlapping and removing noisy examples. These results indicate that the simplicity of KNN can be used as a base for constructing efficient algorithms in machine learning and knowledge discovery. They also show that the selective removal of instances from the majority class is an interesting way to be followed rather than to generate instances to balance datasets. This issue is important nowadays as the datasets are approaching the size of petabytes with big data, and retaining only the representative data can be better than creating more data.

References

[1] Weiss, G.M. and Provost, F. (2001) The Effect of Class Distribution on Classifier Learning: An Empirical Study. Technical Report MLTR-43, Department of Computer Science, Rutgers University, New Brunswick, NJ, USA.

[2] He, H. and Ma, Y. (2013) Imbalanced Learning: Foundations, Algorithms, and Applications. Wiley-IEEE Press, Hoboken, NJ, USA. http://dx.doi.org/10.1002/9781118646106

[3] Japkowicz, N. (2003) Class Imbalances. Are We Focusing on the Right Issue? *Proceedings of the ICML*'2003, *Workshop on Learning from Imbalanced Data Sets II*, Washington DC.

[4] Qiong, G., Cai, Z., Zhu, L. and Huang, B. (2008) Data Mining on Imbalanced Data Sets. *International Conference on Advanced Computer Theory and Engineering*, Phuket, 20-22 December, 1020-1024.

[5] Barandela, R., Sánchez, J.S., García, V. and Rangel, E. (2003) Strategies for Learning in Class Imbalance Problems. *Pattern Recognition*, **36**, 849-851. http://dx.doi.org/10.1016/S0031-3203(02)00257-1

[6] Fawcett, T. (2004) ROC Graphs: Notes and Practical Considerations for Researchers, HP Laboratories.

[7] Chawla, N., Bowyer, K., Hall, L. and Kegelmeyer, W.P. (2002) SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, **16**, 321-357.

[8] Wilson, D.L. (1972) Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Transactions on Systems, Man, and Communications*, **2**, 408-421. http://dx.doi.org/10.1109/TSMC.1972.4309137

[9] Laurikkala, J. (2001) Improving Identification of Difficult Small Classes by Balancing Class Distribution. Technical Report A-2001-2, University of Tampere, Tampere, Finland.

[10] Beckmann, M., De Lima, B.S.L.P. and Ebecken, N.F.F. (2011) Genetic Algorithms as a Pre-Processing Strategy for Imbalanced Datasets. *Proceedings of the* 13*th Annual Conference Companion on Genetic and Evolutionary Computation*, Dublin, 12-16 July 2011, 131-132.

[11] García, S. and Herrera, F. (2009) Evolutionary Undersampling for Classification with Imbalanced Datasets: Proposals and Taxonomy. *Evolutionary Computation*, **17**, 275-396. http://dx.doi.org/10.1162/evco.2009.17.3.275

[12] Hilbert, M. and López, P. (2011) The World's Technological Capacity to Store, Communicate, and Compute Information. *Science*, **332**, 60-65. http://dx.doi.org/10.1126/science.1200970

[13] Wu, X.D., Kumar, V., Quinlan, J.R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S., Zhou, Z.H., Steinbach, M., Hand, D.J. and Steinberg, D. (2007) Top 10 Algorithms in Data Mining. *Knowledge Information Systems*, **14**, 1-37. http://dx.doi.org/10.1007/s10115-007-0114-2

[14] Fix, E. and Hodges, J.L. (1951) Discriminatory Analysis, Nonparametric Discrimination: Consistency Properties. Technical Report 4, USAF School of Aviation Medicine, Randolph Field.

[15] Dasarathy, B.V. (1991) Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques. IEEE Computer Society Press, Los Alamitos.

[16] Duda, R.O., Hart, P.E. and Stork, D.G. (2001) Pattern Classification. 2nd Edition, John Wiley & Sons Ltd., New York, 202-220.

[17] Boriah, S., Chandola, V. and Kumar, V. (2007) Similarity Measures for Categorical Data: A Comparative Evaluation. *Proceedings of the SIAM International Conference on Data Mining*, Minneapolis, 26-28 April 2007, 243-254.

[18] Wilson, D.R. and Martinez, T.R. (1997) Improved Heterogeneous Distance Functions. *Journal of Artificial Intelligence Research*, **6**, 1-34.

[19] Chawla, N.V., Lazarevic, A., Hall, L.O. and Bowyer, K.W. (2003) SMOTEBoost: Improving Prediction of the Minority Class in Boosting. *Proceeding of 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Cavtat-Dubrovnik, 22-26 September 2003, 107-119. http://dx.doi.org/10.1007/978-3-540-39804-2_12

[20] Chen, L., Cai, Z., Chen, L. and Gu, Q. (2010) A Novel Differential Evolution-Clustering Hybrid Resampling Algorithm on Imbalanced Datasets. 3*rd International Conference on Knowledge Discovery and Data Mining*, Phuket, 9-10 January 2010, 81-85.

[21] Han, H., Wang, W.Y. and Mao, B.H. (2005) Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. *Proceedings of International Conference on Intelligent Computing*, Hefei, 23-26 August 2005, 878-887. http://dx.doi.org/10.1007/11538059_91

[22] He, H., Bai, Y. and Garcia, E.A. (2008) ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. *Proceedings of International Joint Conference on Neural Networks*, Hong Kong, 1-8 June 2008, 1322-1328.

[23] Batista, G.E.A.P.A., Prati, R.C. and Monard, M.C. (2004) A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *ACM SIGKDD Explorations Newsletter*, **6**, 20-29. http://dx.doi.org/10.1145/1007730.1007735

[24] Tomek, I. (1976) Two Modifications of CNN. *IEEE Transactions on Systems Man and Communications*, **6**, 769-772. http://dx.doi.org/10.1109/TSMC.1976.4309452

[25] Wang, B.X. and Japkowicz, N. (2004) Imbalanced Data Set Learning with Synthetic Samples. *Proceedings of IRIS Machine Learning Workshop*, Ottawa, 9 June 2004.

[26] Wilson, D.R. and Martinez, T.R. (2000) Reduction Techniques for Instance-Based. *Machine Learning*, **38**, 257-286. http://dx.doi.org/10.1023/A:1007626913721

[27] Van Rijsbergen, C.J. (1979) Information Retrieval. 2nd Edition, Butterworths, Waltham.

[28] Ian, H.W. and Frank, E. (2005) Data Mining: Practical Machine Learning Tools and Techniques. 2nd Edition, Morgan Kaufmann, San Francisco.

[29] Zhang, J.P. and Mani, I. (2003) KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. *Proceeding of International Conference on Machine Learning* (*ICML* 2003), *Workshop on Learning from Imbalanced Data Sets*, Washington DC, 21 August 2003.

[30] Orriols-Puig, A. and Bernadó-Mansilla, E. (2009) Evolutionary Rule-Based Systems for Imbalanced Datasets. *Soft Computing*, **13**, 213-225. http://dx.doi.org/10.1007/s00500-008-0319-7

[31] Blake, C. and Merz, C. (1998) UCI Repository of Machine Learning Databases. Department of Information and Computer Sciences, University of California, Oakland. http://www.ics.uci.edu/~mlearn/~MLRepository.html

[32] Kohavi, R. and Quinlan, J.R. (2002) Decision Tree Discovery. In: Klösgen, W. and Zytkow, J.M., Eds., *Handbook of Data Mining and Knowledge Discovery*, Oxford University Press, New York, 267-276.