DE GRUYTER OPEN

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **17**, No 1
Sofia • 2017 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.1515/cait-2017-0004

On Improving the Classification of Imbalanced Data

Lincy Meera Mathews[1], *Hari Seetha*[2]

[1]*School of Information Technology and Engineering, VIT University, Vellore, Tamil Nadu, India*
[2]*School of Computing Science & Engineering, VIT University, Vellore, Tamil Nadu, India*
Emails: lincymm99@gmail.com hariseetha@gmail.com

***Abstract*:** *Mining of imbalanced data is a challenging task due to its complex inherent characteristics. The conventional classifiers such as the nearest neighbor severely bias towards the majority class, as minority class data are under-represented and outnumbered. This paper focuses on building an improved Nearest Neighbor Classifier for a two class imbalanced data. Three oversampling techniques are presented, for generation of artificial instances for the minority class for balancing the distribution among the classes. Experimental results showed that the proposed methods outperformed the conventional classifier.*

***Keywords*:** *Imbalance data, nearest neighbor classifier, oversampling, synthetic data, Data Mining.*

1. Introduction

A lot of effort is underway to learn from the real world data that are imbalanced in nature through various machine-learning techniques. Imbalanced data occur in many real world applications like fraud detection, medical diagnosis, text categorization and biomedicine. Data is imbalanced when classes are not well-represented by the respective samples equally. Most data mining algorithms assume balanced data. The standard classifiers do well for classes that are well- represented. However, the bias is towards majority class (well-represented classes) when classifiers are applied to imbalanced data. This work proposes to improve the classification of minority instances by modifying the distribution of imbalanced data. Balanced distributions among the classes are formed by generation of synthetic data or by employing algorithmic solutions.

Imbalanced datasets occur due to various reasons. The reasons could be inherent to the domain such as in cases of rare cancer diagnosis, where the incidence of occurrence is very rare or in cases of fraud; the numbers of honest customers are relatively more than the fraud cases. In some cases, the problem of imbalanced data could be due to incomplete collection of data due to privacy or security concerns. A binary imbalanced data comprises of the majority class and the minority class. The minority samples are less represented in the imbalanced data. Consider the cases of rare blood diseases such as sickle cell disease. 95% of the test dataset will reveal

normal blood reports. However, the rest 5% falls over as rare cases of sickle cell diseases. In such scenarios, classifiers are required that will provide a balanced degree of predictive accuracy for the minority and majority set equally. As the minority set has less representation of data, classifiers tend to produce only 0-5% accuracy for minority class, i.e., confirming the presence of sickle cell anaemia. However, the 5% cases are of great interest as it indicates the presence of sickle cell anaemia.

The performance measure for various classifiers is Classification Accuracy (CA). However, in the presence of imbalanced data, this evaluation metric is insensitive to the data distribution. This metric does not indicate the degree of right prediction of minority instances. The CA values are biased towards majority class and do not give an accurate measure of classification of minority tuples. The performance measure, therefore considered here is the Precision, Sensitivity, F-Measure and Geometric mean. All the mentioned measures use parameters that are formed by using a confusion matrix. As shown in the next equations, precision is sensitive to changes in data distribution while recall is not:

$$\text{Precision} = \frac{\text{TP}}{(\text{TP}+\text{FP})}, \tag{1}$$

$$\text{Recall} = \frac{\text{TP}}{(\text{TP}+\text{FN})}. \tag{2}$$

Sensitivity same as Recall, for a 2-class classifier.

F-Measure defines the goodness of the model by including equal weightage for both recall and precision. F-measure provides a high value when both recall and precision are high:

$$\text{F-Measure} = \frac{2\times\text{Recall}\times\text{Precision}}{(\text{Recall}+\text{Precision})}. \tag{3}$$

Geometric mean is a measure that equally focuses on accuracy of both positive and negative instances:

$$\text{GE} = \sqrt{\text{Sensitivity} \times \text{Specificity}}. \tag{4}$$

To assess the performance of various classifiers, AUC is used:

$$\text{AUC} = \frac{(1+\text{TP}_{\text{rate}}-\text{FP}_{\text{rate}})}{2}. \tag{5}$$

The next section covers the related works in the field of mining of imbalanced data.

2. Literature survey

Various solutions are proposed for addressing the imbalance nature among the classes observed over real world data. Researchers have proposed approaches that include altering the distribution of imbalanced dataset to form balanced data, which affects the distribution of data (reduction of skewedness) but not the originality or the quality of data. Forming balanced distribution among imbalanced data is done through some of the below mentioned methods.

Data Level Solutions: This method provides solution that involves altering the distribution of data, which are independent of the classifier. Sampling is one of the methods employed to generate artificial instances. Oversampling, Under sampling and hybrid (oversampling & undersampling) are the different sampling methods [8, 20, 25]. One of the novel works in the area of oversampling is the SMOTE

technique [4, 5]. Informed Under sampling has been used to achieve good sampling results [16, 23, 24].

Cost sensitive techniques: Cost Sensitive methods are being used as an alternative to imbalanced learning domains. Cost matrix is used to account the costs of misclassifying the tuple. Algorithmic changes include algorithms that adjust the costs of the various classes to counter the class imbalance, adjusting the probabilistic estimated and decision threshold at the leaf node of the classification tree [33]. If $C(\text{mi}, \text{ma})$ indicates the cost of misclassifying the majority as minority tuple, then the costs of misclassification of $C(\text{ma}, \text{mi})$ is greater than $C(\text{mi}, \text{ma})$ [31, 29]. Several methods such as AdaC1, AdaC2 and AdaC3 have been proposed to overcome the imbalance distribution [22], which introduced Cost-Sensitive Data space weighting with adaptive Boosting [11]

Algorithmic methods: The classifier models are tuned to learning from the minority class. Existing algorithms are modified or new models are created to adapt to the imbalance nature of the data. It is one of the internal approaches towards solving the imbalance problem [3, 6, 7, 10]. Ensemble approaches (multiple learning algorithms) are also adopted as one of the solutions for classifying imbalanced data [9, 32].

Besides these solutions, researchers have focused on certain characteristics of the imbalanced data [30]. Limited size of training dataset, also expressed as the lack of information available for training of classifier, can also be a factor for misclassification of instances as shown by Viswanath, Murty and Bhatnagar [26], and Seetha, Saravanan and Murty [22]. The high dimensionality of the data in combination with minimal size of the data increases the error in classification by standard models.

The ratio of imbalance affects the performance of the classifier. The higher the imbalance ratio, the minority classes appear as smaller disjuncts [15]. Researchers face a challenge of identifying the disjuncts as positive or rare occurrences or treating those instances as noise [14].

Overlapping of instances among the classes is another factor for reduction in performance by classifiers in imbalanced dataset [19]. The data between the classes cannot be linearly separated. Focus in establishing the relation between overlap and imbalance was studied and inferred by Prati, Batista and Monard [19]. He suggested that the performances of learning algorithms are affected not because of the imbalance but by the degree of overlap between the classes.

Data shift occurs frequently in imbalanced dataset [18]. This is because the training data can be entirely different from test data. As the instances of minority set are minimal, even a single misclassification can bring a drastic decrease in the performance of the classifier [12, 13].

Various techniques have been proposed to reduce the effect of bias of nearest neighbor classifier due to imbalance among the classes. K-NN under sampling methods namely NearMiss-1, NearMiss-2, NearMiss-3 and the most distant method were implemented by Zhang and Mani [29]. NearMiss-1 and NearMiss-2 focus on eliminating samples from majority class based on the average distance to the three farthest samples of minority tuples and on the smallest average distance to the three

farthest minority class samples respectively. Selection of certain number of majority samples for each minority instance is the principle behind NearMiss-3. Several features of the imbalanced dataset such as the under representation of data and high dimensionality of the data [15] can affect the performance of the Nearest Neighbor classifier. Yuxuan and Zhang [28] worked in identifying exemplar minority class training instances and generalizing them to Gaussian balls as concepts for the minority class. In Positive Biased Nearest neighbor classifier, the local neighborhood of query instances is dynamically formed. Classification decision is carefully adjusted based on class distribution in the local neighborhood. This has proved to be efficient for classification of imbalanced data.

This paper studies the effect of nearest neighbor classifier, which is a popular and non-parametric algorithm, on imbalanced data. The focus is on minimizing the bias of nearest neighbor classifier towards majority class by using the oversampling technique. The oversampling method involves generation of synthetic dataset that have the same quality as the original data. Viswanath, Murty and Bhatnagar [26] proposed a novel pattern synthesis method that generated synthetic training instances. The present paper focuses on generating artificial patterns for minority class, using partition based pattern synthesis, which is a novel approach of oversampling the minority class data. The following section discusses the partition method necessary for generation of synthetic instances for the minority class data. Simple_Join, k-JOIN and the BestFit_Synthetic methods for producing synthetic instances are explained briefly. The experimental results are shown followed by conclusion.

3. Pattern generation for minority samples

This paper introduces three techniques of generating synthetic data for minority training instances. The minority synthetic samples are then augmented to the original training data to form a balanced training set. However, synthetic samples can be formed only by partitioning of the original minority dataset. The original minority training dataset is partitioned vertically into blocks. Each block is formed by sub grouping highly correlating features of the dataset into groups. The first section explains the notations and definitions followed in the paper. The feature partition method followed to form blocks is also briefly explained here.

3.1. Notations and definition

The notations used in this paper are defined as follows. Let the imbalanced dataset be DS. TR and TS indicate the training and test set instances respectively. The binary imbalanced dataset comprises of majority and minority tuples, $\mathrm{DS_{ma}}$ and $\mathrm{DS_{mi}}$ respectively. The training instances that belong to minority class are indicated by $\mathrm{TR_{mi}}$. Therefore, $\mathrm{DS_{ma}} \cup \mathrm{DS_{mi}} = \mathrm{DS}$. Let $|\mathrm{TR_{mi}}|$ is denoted as m. Let $\mathrm{DF} = \{F_1, F_2, \ldots, F_p\}$ be the set of features for the dataset where p indicates the total number of features of dataset DS. The set of classes $\mathrm{CS} = \{C_{\mathrm{ma}}, C_{\mathrm{mi}}\}$ indicate the class for majority and minority data respectively. The classes can also be indicated by $\mathrm{CS} = \{-1, +1\}$ for majority and minority tuples respectively. Pattern is defined as

the data instance denoted by x where $x = (v_1, v_2, \dots, v_p)$, such that $x[F_i] = v_i$ for the dataset DS. Let Π denote the set of partitions or blocks for the minority dataset. Thus, $\Pi = \{B_1, B_2, \dots, B_n\}$, where B_i indicates the *i*-th block and $|\Pi| =$ n . The number of partitions for the minority dataset is indicated by n where $1 \leq n \leq p$. If $n == p$, then the number of partitions for the minority class will be same as the degree of the dataset. Let feature set A be $\{F_i, F_j, F_k\}$, for a block where $1 \leq i, j, k \leq p$ and $A \subset$ DF. Lastly, the sub pattern is indicated by X_A and X_A be the projection of data instance x on to the subset of features as defined in feature set A for a block where $X_A \subset x$ and $A \subset$ DF.

3.2. Feature partition method

The minority instances of the training set are vertically partitioned into n number of blocks by applying the feature partition method proposed by V i s w a n a t h, M u r t y and B h a t n a g a r [26]. Each block will contain features that are a subset of the original feature set. The blocks generated will have highly correlating features within the block and the low average correlation between the blocks. This method is used only with numerical features as it is based on pair wise correlation between the features. Correlation coefficients between all features are examined. The features that have less correlation are put into separate blocks whereas features having high correlation are placed in one block. Domain knowledge can also be used to decide on the number of blocks and the cluster of features to represent the block. It was experimented and proved that the classification accuracy tends to fall as the number of blocks increases [26].

4. Synthetic pattern generation from partitioned feature set of minority data

This section aims to describe synthetic pattern generation for minority instances of the training data on which the NN classifier model is built. The balanced training dataset is formed by augmentation of synthetic instances to the original training data. The basic classifier which is tested and experimented against the balanced data is the nearest neighbor algorithm. It is as follows.

Algorithm 1

Nearest_Neigbhor_Classifier:

Input:

a) Training set $\quad TR = TR_{mi} \cup TR_{ma}$

Output:

a) Predicted class, $CS \in \{C_{ma}, C_{mi}\}$

Step 1. Using TR_{mi}, generate synthetic_minority dataset, $DS_{S_{mi}}$.

Step 2. Find *k* greedy neighbors for the test pattern T from $DS_{S_{mi}}$. Let it be denoted by $£_{mi}$.

Step 3. Find *k* greedy neighbors of the test pattern *T* from the majority set TR_{ma}. Let it be denoted by $£_{ma}$.

Step 4. Then find *k* global greedy neighbors of the test instance *T* from the union of $£_{mi} \cup £_{ma}$.

Step 5. Classify *T* to the class based on the majority vote among the *k* global neighbours.

To generate Synthetic minority dataset, $DS_{S_{mi}}$, three methods have been adopted and implemented in this paper. All three methods are categorized as over sampling method. The first algorithm employs the Simple_Join method. This method generates minority tuples by performing Cartesian product of sub pattern between the minority blocks. Deciding the number of partitions or blocks is as explained as above in 3.2. The second algorithm is the k-JOIN. Here, the Cartesian products between the nearest neighbor sub instances from partitions of minority class blocks are computed to produce synthetic samples for a test sample *T*. The third algorithm, BestFit_Synthetic applies the greedy technique to generate new instances, thereby avoiding the join of the sub instances.

4.1. Formal description of the Simple_Join Method

Let $(x_1, x_2, \ldots, x_m)$ are m minority instances from the training data, i.e., $|TR_{mi}| = m$. Let $\Pi = \{B_1, B_2, \ldots, B_n\}$ be the vertical (Feature split) partition of training samples of the minority class. The majority training instances are denoted by TR_{ma}. Then $X_{iB_1}, X_{iB_2}, \ldots, X_{iB_n}$, are sub-instances of sample x_i belonging to each block of the minority class of block B_j where $1 \leq i \leq m$ and $1 \leq j \leq n$. Cartesian product of the minority sub tuples between all the blocks are performed, thereby producing m^n synthetic tuples where m is the number of minority instances and n is the total number of blocks. These synthetic samples are added to the original training data to reduce the ratio of imbalance between the majority and minority classes. However, in certain datasets, the synthetic minority instances generated after the join exceed the number of existing majority samples. To form a balanced representation of samples within both the classes, clustering is then performed on the synthetic minority instances generated. The number of clusters formed will be almost less than or equal to the number of majority tuples present in the training set. The centroid of each cluster is accepted as the synthetic minority tuple.

Algorithm 2

Simple_Join:

Input:

a) Partitioned Minority Data Blocks $\Pi = \{B_1, B_2, \ldots, B_n\}$ from DS_{mi} containing *m* sub instance in each block

Output:

a) synthetic_minority dataset, $DS_{S_{mi}}$

For all blocks: from 1 to *n* do

Find Cartesian product of all tuples between blocks to generate

S_{mi}, where $S_{mi} = B_1 \times B_2 \times \ldots \times B_n$, where n indicates the number of blocks

Endfor

If $|S_{mi}| > |DS_{ma}|$, then

Begin

Choose $k = |DS_{ma}|$

Perform k-means clustering on S_{mi}
Select centroid of each cluster. Let it be denoted as C_i
synthetic_minority Dataset, $\text{DS}_{S_{\text{mi}}} = \cup\ C_i$
End
Else
synthetic_minority Dataset, $\text{DS}_{S_{\text{mi}}} = S_{\text{mi}}$
Endif

4.1.1. Space and time requirement for Simple_Join:

Space requirement is $O(m^n)$, where *m* is the size of the original minority training set size, and *n* is the number of blocks. Time requirement of the method for the method is $O(m^n)$. An example to indicate Simple_Join is shown in Fig 1. The illustration is done assuming two block partition.

Set, DF = $\{F_1, F_2, F_3, F_4\}$
$\text{DS}_{\text{mi}} = [t_1(a,b,c,d), t_2(p,q,r,s), t_3(l,m,n,o), t_4(w,x,y,z)]$
$\Pi = \{B_1, B_2\}$, where B_1={ F_1, F_3} and B_2={ F_2, F_4}
All original minority tuples are considered and split into 2 blocks.
Therefore DS_{mi} of B_1= $[(a,*,c,*), (p,*,r,*), (l,*,n,*), (w,*,y,*)]$
DS_{mi} of B_2= $[(*,b,*,d), (*,q,*,s), (*,m,*,o), (*,x,*,z)]$
Simple_Join

Fig 1. Illustration of Simple_Join

4.2. Formal description of the k-JOIN Method

The minority data is vertically partitioned into blocks, $\Pi = \{B_1, B_2, \ldots, B_n\}$. For a given test tuple *T*, the k-nearest neighbor sub instances from blocks B_1 to B_n are found. The nearest neighbors for a test sample *T* are found by splitting the test sample with respect to the block features. The distance measure is applied to find the k-nearest neighbors for the sample *T* with block B_i. The Cartesian products of the k-nearest neighbors sub instances are found from the n blocks, and are augmented to DS_{mi} to form $\text{DS}_{S_{\text{mi}}}$.

Therefore, $\left|\text{DS}_{S_{\text{mi}}}\right| = k^n$, where *k* is the number of nearest neighbors and *n* is the number of blocks or partition for the minority class. Algorithm 3 describes the synthetic dataset generated by k-JOIN.

Synthetic instances created during training phase

$$\text{DS}_{S_{\text{mi}}} = B_1 \times B_2 = \left\{ \begin{matrix} (a,b,c,d) \\ (a,q,c,s) \\ a,m,c,o), \\ (a,x,c,z), \\ (p,b,r,d), \ldots \\ (w,x,y,z) \end{matrix} \right\},$$

$\left|\text{DS}_{S_{\text{mi}}}\right| = |\text{DS}_{\text{mi}}|^2$ as $|\Pi| = 2 = 4^2 = 16$; synthetic instances are produced.

Algorithm 3

k-JOIN:

Input:

a) Partitioned Minority Training data, $\mathrm{DS_{mi}}$, into Blocks $\Pi = \{B_1, B_2, \ldots, B_n\}$

b) Test Set, TS

Output:

a) Synthetic_Minority Dataset, $\mathrm{DS}_{S_{\mathrm{mi}}}$

For each test sample *T* do

For *j*: 1 to *n* do

For each block $B_j \, \varepsilon \, \Pi$ do

Let the nearest *k* neighbors of T_{B_j} in $\mathrm{DS}_{B_j}$with respect to subset of features in B_j be N_j,

where $N_j = \left\{X_{B_j}^{j1}, X_{B_j}^{j2}, \ldots, X_{B_j}^{jk}\right\}$

Endfor

Endfor

Find the Cartesian product $N = N_1 \times N_2 \times \ldots \times N_n$, where $|N| = k^n$

Endfor

synthetic_Minority Dataset, $(\mathrm{DS}_{S_{\mathrm{mi}}}) = N \cup \mathrm{DS_{mi}}$

Space and time requirement for k-JOIN. Space requirement is $O(k^n)$, where *k* indicates the number of nearest neighbors and *n* is the number of blocks. Time requirement of the method is $O(k^n)$. The working example of k-JOIN is shown in Fig. 2.

Set, DF = $\{F_1, F_2, F_3, F_4\}$
$\mathrm{DS_{mi}}$=$[t_1(a,b,c,d), t_2(e,f,g,h), t_3(p,q,r,s), t_4(l,m,n,o), t_5(w,x,y,z)]$
$\Pi = \{B_1, B_2\}$, where B_1= { F_1, F_3} and B_2= $\{F_2, F_4\}$
Therefore,
Let the 3-nearest neighbor (wrt features of B_1) for test sample *T* be
N_1= $[t_3(p,*,r,*), t_4(l,*,n,*), t_5(w,*,y,*)]$
And from
N_2=$[t_1(*,b,*,d), t_3(*,q,*,s), t_4(*,m,*,o)]$ from Block B_1 and B_2 respectively.

k-JOIN

Synthetic instances created during run phase

$$\mathrm{DS}_{S_{\mathrm{mi}}} = N_1 \times N_2 = \left\{\begin{matrix}(p,b,r,d)\\(p,q,r,s)\\(p,m,r,o),\\(l,b,n,d),\\(l,q,n,s), \ldots\\(w,m,y,o)\end{matrix}\right\}$$

$\left|\mathrm{DS}_{S_{\mathrm{mi}}}\right| = |3|^2$ as the $|\Pi| = 2$ and *k*=3
= 3*3= 9; synthetic instances are produced

BestFit_Synthetic:

Synthetic tuples created during run phase

$$\mathrm{DS}_{S_{\mathrm{mi}}} = N_1 + N_2 = \left\{\begin{matrix}(p,b,r,d),\\(l,q,n,s)\\w,m,y,o)\end{matrix}\right\}$$

$\left|\mathrm{DS}_{S_{\mathrm{mi}}}\right| = k = 3$; Synthetic instances are produced

Fig. 2. Illustration of k-JOIN and BestFit_Synthetic

4.3. Formal description of the BestFit_Synthetic

A greedy technique is performed here to generate the synthetic instances. The synthetic instances are generated by combining the *i*-th sub pattern of *k* nearest samples of all blocks for a given test sample. Initially, the *k* nearest neighbors for the test sample *T*, with respect to sub features from each block is found. The union with respect to each row of the *k* nearest neighbor of all blocks forms the synthetic set. The union execution is by joining each row tuple of the nearest neighbor of each block respectively.

The number of synthetic instances produced is equal to *k* for each test instance.

Algorithm 4

BestFit_Synthetic:

Input:

Partitioned Minority Training Data $\mathrm{TR_{mi}}$, into Blocks $\Pi = \{B_1, B_2, \dots, B_n\}$ of the minority tuples synthetic_minority dataset, $\mathrm{DS}_{S_{\mathrm{mi}}} = \{\mathrm{TR_{mi}}\}$

Test Set, TS

Output: synthetic_minority Dataset, $\mathrm{DS}_{S_{\mathrm{mi}}}$

For each test pattern T do

For each block *j*: from 1 to *n* do

For feature set of B_j

Find *k* nearest neighbors of T, with respect to features in TR_{B_j}

Let the *k* nearest neighbors be $N_j = \left\{X_{B_j}^{j1}, X_{B_j}^{j2}, \dots, X_{B_j}^{jk}\right\}$, where $X_{B_j}^{ji}$ is the *i*-th nearest tuple in *j*-th block

EndFor

EndFor

EndFor

The Synthetic set generated will be as follows:

$$\mathrm{DS}_{S_{\mathrm{mi}}} = \left\{\left(X_{B_1}^{11}, X_{B_2}^{21}, \dots, X_{B_n}^{n1}\right), \left(X_{B_1}^{12}, X_{B_2}^{22}, \dots, X_{B_n}^{n2}\right), \dots, \left(X_{B_1}^{1k}, X_{B_2}^{2k}, \dots, X_{B_n}^{nk}\right)\right\} \cup \mathrm{TR_{mi}}$$

Space and time requirement for BestFit_Synthetic. Space and time requirements of the method are *O*(*k*) as the space taken by the synthetic samples generated is equal to the number of nearest neighbor, i.e k.

5. Experimental results

Experimental studies are described here.

5.1. Datasets description

Imbalanced datasets from the UCI machine learning [8] repository and Keel repository [1, 2] were chosen. The chosen datasets are indicated in Table 1. The attributes and the number of instances of majority and minority classes are

mentioned. In both the repositories, the datasets were chosen and subjected to 10 cross validations. For experimentation purpose, each set is used as test set. However, in real time, real times test samples can be used to generate the synthetic instances.

All the datasets have only numeric valued features and continuous value attributes. These datasets are classified mainly into two classes- majority class and minority class. The majority class has good data representation whereas minority dataset are outnumbered and less represented. The majority class is the negative class (–1) whereas the minority class is the positive class (+1).

The experimentation has been done with a split of two blocks. Experimentation results by Viswanath, Murty and Bhatnagar [26], shows that when the datasets are divided into more than two or three blocks, classification does not make much improvement. This is because even though the generation of synthetic tuples increases, the synthetic distribution for larger values of *n* will greatly vary when compared to the original data.

5.2. Results

Table 1 describes the benchmarked imbalanced datasets trained by the k-NN classifier. For each dataset, we have presented values for various training samples achieved under k-NN classifier. The training samples include original samples and synthetic samples generated by Oversampling, Simple_Join, k-JOIN and BestFit_Synthetic. As shown from Tables 2-8, each dataset with their performance measures have been indicated. The k parameters for the nearest neighbor classifier are experimented for 1, 3, 5 and 7. Different imbalance ratios have been taken into consideration. 10-fold cross validation was performed to validate the results [25]. For different datasets, values attained for Sensitivity, F-Measure, Geometric Mean and AUC are mentioned in their respective table. Tables 2-8 describes the results obtained under various measures for samples generated using different techniques.

Table 1. Description of Data Sets

Name	Number of classes	Examples (+, –)	Number of attributes
Vehicle	2	212:634	18
Haberman	2	81:225	3
LiverPatient	2	268:500	10
Vertabre	2	100:210	6
Prima_Indian_Diabetes	2	268:500	8
Blood Transfusion	2	178:570	4
Spectrometer	2	45:486	93

Table 2. Comparison between measures for vehicle

Measure	Sampling Technique (Vehicle)	$k = 1$	$k = 3$	$k = 5$	$k = 7$
Precision	Original Data	43.13	48.78	54.21	52.89
	Oversampling	83.39	85.71	89.26	89.46
	Simple_Join	85.75	81.93	80.91	80.5
	k-JOIN	94.73	92.11	90.5	89.52
	BestFit_Synthetic	**94.97**	**92.67**	**91.05**	**89.69**
Sensitivity	Original Data	37.25	37.27	33.51	25.93
	Oversampling	79.95	80.26	79.47	77.42
	Simple_Join	96.06	92.27	90.69	87.54
	k-JOIN	100	99.91	99.91	99.87
	BestFit_Synthetic	99.27	99.53	99.7	99.74
F-Measure	Original Data	39.97	42.26	41.42	34.8
	Oversampling	81.63	82.90	84.08	83.00
	Simple_Join	90.61	86.79	85.52	83.88
	k-JOIN	97.29	95.86	94.97	94.41
	BestFit_Synthetic	**97.07**	**95.98**	**95.18**	**94.45**
Geometric Mean	Original Data	56.44	58.81	52.84	48.73
	Oversampling	81.92	83.32	79.67	73.80
	Simple_Join	89.63	82.65	78.25	75.35
	k-JOIN	89.14	83.97	79.70	75.71
	BestFit_Synthetic	**89.42**	**85.61**	**84.37**	**82.93**
AUC	Original Data	60.13	61.67	60.86	59.68
	Oversampling	82.01	83.43	80.93	78.14
	Simple_Join	89.26	84.69	81.92	79.01
	k-JOIN	90.21	84.33	81.12	78.60
	BestFit_Synthetic	**90.27**	**84.69**	**83.74**	**82.56**

Table 3. Comparison between measures for Haberman

Measure	Sampling Technique (Haberman)	$k = 1$	$k = 3$	$k = 5$	$k = 7$
Precision	Original Data	39.40	43.96	46.96	52.33
	Oversampling	75.89	77.86	80.59	81.82
	Simple_Join	51.63	65.39	68.60	71.27
	k-JOIN	88.24	87.05	85.60	85.47
	BestFit_Synthetic	**91.43**	**92.22**	**91.28**	**90.80**
Sensitivity	Original Data	39.30	34.58	29.72	23.47
	Oversampling	80.81	77.25	75.42	76.28
	Simple_Join	44.05	56.44	60.05	64.94
	k-JOIN	85.63	90.01	92.36	94.49
	BestFit_Synthetic	**87.76**	**94.95**	**96.07**	**96.18**
F-Measure	Original Data	39.35	38.71	36.40	32.40
	Oversampling	78.27	77.55	77.92	78.95
	Simple_Join	47.54	60.59	64.04	65.44
	k-JOIN	86.92	88.51	88.85.	89.76
	BestFit_Synthetic	**89.55**	**93.56**	**93.61**	**93.41**
Geometric Mean	Original Data	54.37	49.91	47.38	44.02
	Oversampling	67.32	63.32	68.36	67.18
	Simple_Join	50.54	62.41	65.76	67.42
	k-JOIN	68.04	64.28	68.88	67.33
	BestFit_Synthetic	**76.49**	**80.04**	**77.88**	**76.62**
AUC	Original Data	58.79	58.74	57.82	56.64
	Oversampling	64.59	66.66	67.36	64.26
	Simple_Join	51.51	59.62	64.68	64.73
	k-JOIN	72.20	69.14	66.61	64.76
	BestFit_Synthetic	**77.49**	**81.03**	**79.29**	**78.61**

Table 4. Comparison between measures for Liver Patient data

Measure	Sampling Technique (LiverPatient)	$k = 1$	$k = 3$	$k = 5$	$k = 7$
Precision	Original Data	37.15	36.64	37.65	36.49
	Oversampling	73.30	77.87	79.23	79.52
	Simple_Join	78.08	71.59	67.41	65.33
	k-JOIN	93.73	91.66	90.09	88.77
	BestFit_Synthetic	**94.18**	**92.05**	**90.36**	**89.07**
Sensitivity	Original Data	39.81	34.26	32.42	30.00
	Oversampling	77.57	74.94	74.94	73.74
	Simple_Join	95.21	84.63	82.96	81.04
	k-JOIN	99.89	99.78	99.29	98.63
	BestFit_Synthetic	**99.29**	**99.51**	**99.40**	**99.13**
F-Measure	Original Data	38.44	35.41	34.84	32.93
	Oversampling	75.37	76.38	77.02	76.52
	Simple_Join	85.80	77.57	74.38	72.34
	k-JOIN	96.71	95.54	94.46	93.44
	BestFit_Synthetic	**96.67**	**95.63**	**94.66**	**93.83**
Geometric Mean	Original Data	52.72	50.22	48.99	45.72
	Oversampling	74.38	76.43	70.98	67.00
	Simple_Join	82.75	74.23	69.90	67.40
	k-JOIN	83.81	77.21	71.50	66.03
	BestFit_Synthetic	**84.95**	**78.5**	**72.53**	**67.39**
AUC	Original Data	57.11	56.25	54.86	54.85
	Oversampling	74.59	76.66	74.36	70.26
	Simple_Join	82.08	75.48	70.79	71.29
	k-JOIN	86.33	79.67	75.63	72.15
	BestFit_Synthetic	85.77	**80.42**	**76.63**	71.21

Table 5. Comparison between measures for Vertabre

Measure	Sampling Technique (Vertabre)	$k = 1$	$k = 3$	$k = 5$	$k = 7$
Precision	Original Data	72.37	71.45	76.44	79.23
	Oversampling	88.17	87.57	86.36	87.51
	Simple_Join	80.06	78.65	79.29	79.37
	k-JOIN	95.41	94.04	94.17	93.92
	BestFit_Synthetic	**96.17**	**94.53**	**93.89**	93.41
Sensitivity	Original Data	77.00	76.00	72.00	76.00
	Oversampling	89.45	87.57	89.45	89.45
	Simple_Join	81.9	85.71	88.1	90
	k-JOIN	99.73	99.73	99.27	99.27
	BestFit_Synthetic	**99.82**	**100**	**99.73**	**99.82**
F-Measure	Original Data	74.62	73.66	74.15	77.58
	Oversampling	88.81	87.40	87.88	88.47
	Simple_Join	80.97	82.03	83.46	84.35
	k-JOIN	97.52	96.8	96.65	96.52
	BestFit_Synthetic	**97.96**	**97.19**	**96.72**	**96.51**
Geometric Mean	Original Data	81.14	80.26	79.33	82.56
	Oversampling	82.40	80.93	78.48	77.90
	Simple_Join	80.43	80.49	81.71	82.25
	k-JOIN	**86.23**	**81.26**	**81.68**	**80.81**
	BestFit_Synthetic	**88.68**	**83.18**	80.55	78.79
AUC	Original Data	81.33	79.59	81.29	83.05
	Oversampling	82.54	80.12	80.58	78.06
	Simple_Join	80.71	81.90	80.95	82.38
	k-JOIN	86.20	83.05	82.78	82.97
	BestFit_Synthetic	**89.39**	**84.29**	**82.86**	81.67

Table 6. Comparison between measures for Prima_Indian_Diabetes

Measure	Sampling Technique (Prima_Indian_Diabetes)	$k = 1$	$k = 3$	$k = 5$	$k = 7$
Precision	Original Data	55.36	57.32	61	61.84
	Oversampling	75.29	77.15	79.28	80.87
	Simple_Join	65.99	68.54	69.43	69.41
	k-JOIN	92.49	90.58	89.87	89.78
	BestFit_Synthetic	**94.09**	**91.97**	**90.95**	**90.32**
Sensitivity	Original Data	53.32	53.68	53.3	54.06
	Oversampling	74.54	74.75	74.94	73.34
	Simple_Join	63.6	73.4	74.8	75.6
	k-JOIN	97.52	98.71	98.91	98.91
	BestFit_Synthetic	**98.78**	**99.08**	**99.12**	**99.25**
F-Measure	Original Data	54.32	55.44	56.89	57.69
	Oversampling	74.91	75.93	72.05	72.47
	Simple_Join	64.77	70.89	72.01	72.37
	k-JOIN	94.94	94.47	94.17	94.13
	BestFit_Synthetic	**96.38**	**95.4**	**94.86**	**94.58**
Geometric Mean	Original Data	63.49	64.55	65.5	66.22
	Oversampling	64.90	65.96	67.18	66.75
	Simple_Join	64.89	69.26	64.18	65.66
	k-JOIN	71.9	62.23	68.06	67.54
	BestFit_Synthetic	**79.07**	**69.61**	70.42	**70.44**
AUC	Original Data	63.28	65.07	67.05	68.42
	Oversampling	65.07	66.18	61.47	63.17
	Simple_Join	63.80	71.00	70.00	66.38
	k-JOIN	76.23	69.35	67.72	66.99
	BestFit_Synthetic	**80.39**	**73.87**	**70.14**	**70.80**

Table 7. Comparison between measures for Blood_Transfusion

Measure	Sampling Technique (Blood_Transfusion)	$k = 1$	$k = 3$	$k = 5$	$k = 7$
Precision	Original Data	36.08	39.03	37.66	34.47
	Oversampling	68.71	76.53	79.91	80.04
	Simple_Join	77.32	81.72	80.97	78.88
	k-JOIN	89.67	89.09	88.47	87.56
	BestFit_Synthetic	**92.52**	**94.31**	**93.38**	**91.93**
Sensitivity	Original Data	36.54	33.73	29.22	26.34
	Oversampling	73.54	73.66	70.15	70.33
	Simple_Join	73.33	74.74	74.04	74.04
	k-JOIN	78.3	82.74	85.04	86.26
	BestFit_Synthetic	**82.48**	**84.68**	**86.21**	**87.39**
F-Measure	Original Data	36.08	39.03	37.66	34.47
	Oversampling	75.81	78.94	77.03	74.18
	Simple_Join	75.27	78.07	77.35	76.38
	k-JOIN	83.6	85.8	86.72	86.91
	BestFit_Synthetic	**87.21**	**89.23**	**89.65**	**89.6**
Geometric Mean	Original Data	52.99	53.63	50.19	47.92
	Oversampling	71.81	74.75	71.81	69.84
	Simple_Join	73.68	73.64	72.87	70.81
	k-JOIN	75.4	78.28	77.29	76.36
	BestFit_Synthetic	**79.61**	**83.39**	**82.28**	**80.03**
AUC	Original Data	57.99	60.90	59.69	58.60
	Oversampling	72.88	78.94	70.03	70.18
	Simple_Join	73.88	73.51	72.60	71.37
	k-JOIN	75.21	78.11	78.62	77.16
	BestFit_Synthetic	**80.82**	**83.69**	**82.57**	**80.54**

Table 8. Comparison between measures for Spectrometer

Measure	Sampling Technique (Spectrometer)	$k = 1$	$k = 3$	$k = 5$	$k = 7$
Precision	Original Data	91.67	92.67	93.5	93.5
	Oversampling	98.58	98.17	98.17	98.07
	Simple_Join	99.4	98.61	98.81	98.79
	k-JOIN	99.6	98.6	98.07	97.86
	BestFit_Synthetic	**99**	**98.61**	**98.42**	**98.19**
Sensitivity	Original Data	77	72.5	70.5	68
	Oversampling	97.94	97.93	97.53	97.32
	Simple_Join	100	98.79	97.57	97.37
	k-JOIN	100	98.99	97.58	97.17
	BestFit_Synthetic	98.78	98.98	**99.19**	**98.38**
F-Measure	Original Data	83.7	81.35	80.39	78.74
	Oversampling	98.54	98.45	98.24	98.34
	Simple_Join	99.29	98.58	98.68	98.05
	k-JOIN	99.6	98.39	97.68	97.78
	BestFit_Synthetic	98.89	**98.8**	**98.8**	**98.28**
Geometric Mean	Original Data	86.72	84.19	83.12	81.05
	Oversampling	97.55	97.45	97.24	97.35
	Simple_Join	99.28	98.55	98.65	98.02
	k-JOIN	99.59	98.56	98.06	97.86
	BestFit_Synthetic	**99.87**	**98.77**	**98.77**	**98.25**
AUC	Original Data	87.09	86.19	85.19	84.09
	Oversampling	98.56	98.46	98.25	98.35
	Simple_Join	99.39	99.18	98.46	97.33
	k-JOIN	99.38	98.47	97.97	98.07
	BestFit_Synthetic	**99.88**	**98.67**	**98.36**	**98.46**

5.3. Discussion

Observation 1. *As seen, the standard nearest neighbor shows the poorest performance when trained on original imbalanced training set for all datasets. Sensitivity values indicate the high misclassifications of minority instances*. As the value of k increases, the recognition rate of positive samples decreases. This effect is caused due to the fact that as the value of k increases, more majority samples will be inducted during decision making process at the testing phase.

Observation 2. *For all datasets, the k-NN (for all values of k) has shown commendable performance when trained on samples formed by BestFit_Synthetic than Oversampling, Simple_Join or k-JOIN. This method in terms of space and time utility, also occupies the least when compared to the other methods introduced*. The number of synthetic samples generated by BestFit_Synthetic is equal to the value of k chosen. The k value chosen however chosen must be sufficient enough to balance the training set. The F-Measure, Geometric Mean and sensitivity also indicates the goodness of the algorithm.

Observation 3. *Certain datasets like Haberman and Vertabre, Oversampling showed better performance than artificial samples generated by Simple_Join. This performance is seen due to the presence of redundant samples in the data itself. Hence, Random Oversampling enhanced the occurrence of coincident samples. This can further lead to overfitting*. In contrast, it has to be noted that synthetic samples generated by Oversampling did not prove to be as efficient as samples generated from our three techniques for Data such as in LiverPatient, Vehicle, etc.

Observation 4. *In vertebra and vehicle, the performance of k-NN classifier from samples generated by BestFit_Synthetic shows only a slight difference in terms of sensitivity when compared with other sampling methods. Nevertheless, its best to employ BestFit_Synthetic, for it requires less execution time and space*. G-Mean measure is to maximize the accuracy on both the classes while these accuracies are still balanced. G-mean of BestFit_Synthetic has shown competitive performance when compared to all other implemented methods. F- Measure, which focuses on the goodness of the algorithm, clearly indicates high measure performance by our algorithms compared to Nearest Neighbor algorithm.

Observation 5. *BestFit_Synthetic forms k greedy-neighbors synthetic representatives for the minority class. The implementation advantage with this method when compared to k-JOIN is that the BestFit_Synthetic avoids Cartesian product. It forms synthetic samples by concatenating i-th nearest neighbor between the blocks.* The AUC measures also indicate that the greedy technique (BestFit_Synthetic) is better than the all other methods for all datasets. Classification of both positive and negative tuples also improves in BestFit_Synthetic.

Observation 6. *The visible better performance by the BestFit_Synthetic is the fact that the synthetic samples generated minimizes over fitting***.** Limited synthetic instances are produced due to the greedy method involved in generating the instances. As a result, classification of minority instances increases; the misclassification of majority samples also reduces. It is seen that as more synthetic samples are generated, the samples tend to over generalize the minority class, without keeping in view that these synthetic samples could be seen as noise by the majority instances. This is also highly viewed as a disadvantage of oversampling. Overfitting is considered as one of the prominent disadvantage of oversampling.

To understand it, we see that the number of synthetic samples generated under Simple_Join is m^n. However, in *k*-JOIN, the number of produced samples is k^n for each sample. Here, the number of synthetic samples is reduced drastically. This creates more separation between the minority and majority class as the synthetic samples produced are from join of minority training instances that are closer to the test instance *T*. The distance of the decision region between the majority and minority classes increases. A likely minority or majority test instance will pick sub instances from the blocks that are closer to the test sub pattern.

In BestFit_Synthetic, *k* synthetic samples only are generated using the greedy method for a test sample. Hence, the samples are relatively very much similar to the test instances. The distribution of the synthetic samples is also more similar to the original training instances. The misclassification of majority instances will be the least as the decision region between the majority samples and minority samples will be maximum and distinct.

For concluding the discussion section, we have shown three techniques that provide quality performances to improve the bias of the classifier towards minority data. It is shown a considerable improvement of the performance of the *k*-NN classifier by providing *k* samples for each test instance through BestFit_Synthetic.

5.4. Limitations and future work

1. All methods depend on correlation coefficient. Therefore, data that have numerical attributes only can employ these algorithms. Domain knowledge can be an alternative to split attributes into blocks. Suitable methods for data sets having categorical attributes would be further explored as future work.

2. The other limitation is that this method is applied only if the features can be correlated or grouped into blocks.

3. As for big data with hundreds of attributes, these methods are feasible as originally the methods involved were to address the curse of dimensionality problem. However, improvising space and time constraints can be a possible future work.

Description of the pros and cons of each technique is as mentioned in Table 9.

Table 9. Pros and Cons of methods used to generate Synthetic samples

Method adopted	Pros	Cons
Simple_Join	Synthetic Instances are formed before the test phase Run time for classifying the test instance is minimal	The number of instances formed are very huge in number leading to wastage of space
k-JOIN	Synthetic Instances are formed during the test phase as per the test instance. Space requirement is minimal Instances formed are less than Simple_Join	Test time phase increases as the synthetic instances formed during run time
BestFit_Synthetic:	Run time formation of instances The number of synthetic instances formed is the least in this method. Hence, the space required is the least even when compared with oversampling	Test time phase increases as the synthetic instances formed during run time but comparatively less time compared to *k*-JOIN

6. Conclusions

Three techniques for producing synthetic samples for imbalanced data are proposed and implemented in this paper. The proposed oversampling algorithms work towards forming an equal distribution of minority and majority data. This is implemented by the production of artificial samples for minority class. All three methods are based on splitting the minority instances into two or more blocks. The blocks are formed by studying the problem domain or by feature set partition method. The features presented in each block are highly correlated to each other whereas inter block correlations between features are minimal.

Performances of k-NN indicates comparison with original samples and synthetic samples generated by oversampling, Simple_Join, k-JOIN and BestFit_Synthetic. All the four methods outperformed the standard nearest neighbor algorithm proven by using the standard measures of Geometric mean, precision, F-Measure, AUC and Sensitivity. The BestFit_Synthetic produces visibly competitive results when compared k-JOIN, Simple_Join and standard Nearest Neighbor algorithm**.** The accuracy has been improved for both minority and majority class, when compared to nearest neighbor algorithm. This method is also applicable to all data having more than one minority class by generating samples for all minority classes.

References

1. Alcalá-Fdez, J., L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, F. Herrera. KEEL: A Software Tool to Assess Evolutionary Algorithms to Data Mining Problems. – Soft Computing, Vol. **13**, 2009, No 3, pp. 307-318.
2. Alcalá-Fdez, J., A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, 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**, 2011, No 2-3, pp. 255-287.
3. Barandela, R., J. S. Sanchez, V. Garcia, E. Rangel. Strategies for Learning in Class Imbalance Problems. – Pattern Recogn., Vol. **36**, 2003, No 3, pp. 849-851.
4. Krawczyk, B. Learning from Imbalanced Data: Open Challenges and Future Directions. – Progress in Artificial Intelligence, Vol. **5**, November 2016, No 4, pp. 221-232.
5. Chawla, N. V., K.,W. Bowyer, L.,O. Hall, W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-Sampling Technique. – J. Artificial Intelligence Research, Vol. **16**, 2002, pp. 321-357.
6. Chawla, N. V., N. Japkowicz, A. Kolcz. Editorial: Special Issue on Learning from Imbalanced Data Sets. – ACM SIGKDD Explorations Newsletter, Vol. **6**, 2004, No 1, pp. 1-6.
7. Ramentol, E., S. Vluymans, N. Verbiest, Y. Caballero, R. Bello, C. Cornelis, F. Herrera. IFROWANN: Imbalanced Fuzzy-Rough Ordered Weighted Average Nearest Neighbor Classification. – IEEE Transactions on Fuzzy Systems, Vol. **23**, October 2015, No 5, pp. 1622-1637.
8. Frank, A., A. Asuncion. UCI machine learning repository. 2010. **http:// archive.ics.uci.edu/ml**
9. Galar, M., A. Fernando, E. Barrenechea, H. Business, F. Herrera. A Review on Ensembles for the Class Imbalance Problem. – IEEE Transactions on Systems, Man, and Cybernetics. Part C: Applications and Review, Vol. **42**, 2012.
10. Garcia-Pedrajas, N., J. Perez-Rodriguez, M. Garcia-Pedrajas, D. Ortiz-Boyer, C. Fyfe. Class Imbalance Methods for Translation Initiation Site Recognition in DNA Sequences. – Knowl. Based Syst., Vol. **25**, 2012, No 1, pp. 22-34.
11. Guo, H., H. Viktor. Learning from Imbalanced Data Sets with Boosting and Data Generation: The Databoost-im Approach. – SIGKDD Explorations, Vol. **6**, 2004, pp. 30-39.
12. Lee, J., D.-W. Kim. Mutual Information-Based Multi-Label Feature Selection Using Interaction Information. – Expert Systems with Applications, Vol. **42**, March 2015, No 4, pp. 2013-2025.
13. Jain, A., B. Chandrasekharan. Dimensionality and Sample Size Considerations in Pattern Recognition Practice. – In: P. Krishnaiah, L. Kanal, Eds. Handbook of Statistics. Vol. **2**. North Holland, 1982, pp. 835-855.
14. Jo, T., N. Japkowicz. Class Imbalances Versus Small Disjuncts. – ACM SIGKDD, Vol. **6**, 2004, No 1, pp. 40-49
15. Sáez, J. A., B. Krawczyk, M. Woźniak. Analyzing the Oversampling of Different Classes and Types of Examples in Multi-Class Imbalanced Datasets. – Pattern Recogn., Vol. **57**, 2016, pp. 164-178.
16. Peng, L., et al. Imbalanced Traffic Identification Using an Imbalanced Data Gravitation-Based Classification Model. – Computer Communications, 2016, pp. 347-373.
17. Moreno-Torres, J. G., F. Herrera. A Preliminary Study on Overlapping and Data Fracture in Imbalanced Domains by Means of Genetic Programming-Based Feature Extraction. – In: 10th International Conference on Intelligent Systems Design and Applications (ISDA'2010), 2010, pp. 501-506,
18. Moreno-Torres, J. G., T. Raeder, R. Alaíz-Rodríguez, N. V. Chawla, F. Herrera. A Unifying View on Dataset Shift in Classification. – Pattern Recogn., Vol. **45**, 2012, No 1, pp. 521-530.

19. P r a t i, R. C., G. E. A. P. A. B a t i s t a, M. C. M o n a r d. Class Imbalances Versus Class Overlapping: An Analysis of a Learning System Behavior. – In: R. Monroy, G. Arroyo-Figueroa, L. E. Sucar, H. Sossa, Eds. MICAI 2004. LNCS (LNAI). Vol. **2972**. Heidelberg, Springer, 2004, pp. 312-321.
20. Y i n, Q.-Y., J.-S. Z h a n g, C.-X. Z h a n g, N.-N. J i. A Novel Selective Ensemble Algorithm for Imbalanced Data Classification Based on Exploratory Undersampling. – In: Hindawi Publishing Corporation Mathematical Problems in Engineering. Vol. **2014**. Article ID 358942, 14 p.
21. S a t u l u r i, N., M. R. K u p p a. A Novel Class Imbalance Learning Using Intelligent Under-Sampling. – International Journal of Database Theory and Application, Vol. **5**, 2012, pp. 25-35.
22. S e e t h a, H., R. S a r a v a n a n, M. N. M u r t y. Pattern Synthesis Using Multiple Kernel Learning for Efficient SVM Classification. – Cybernetics and Information Technologies, Vol. **12**, 2012, No 4, pp. 77-94.
23. S u n, Y., M. S. K a m e l, A. K. C. W o n g, Y. W a n g. Cost-Sensitive Boosting for Classification of Imbalanced Data. – Pattern Recogn., Vol. **40**, 2007, pp. 3358-3378.
24. T a h i r a, M. A., J. K i t t l e r a, F. Y a n. Inverse Random under Sampling for Class Imbalance Problem and its Application to Multi-Label Classification. – Pattern Recogn., Vol. **45**, 2012, No 10, pp. 3738-3750.
25. L ó p e z, V., A. F e r n á n d e z, F. H e r r e r a. On the Importance of the Validation Technique for Classification with Imbalanced Datasets: Addressing Covariate Shift when Data is Skewed. – Information Sciences, Vol. **257**, February 2014, pp. 1-13.
26. V i s w a n a t h, P., M. N. M u r t y, S. B h a t n a g a r. Partition Based Pattern Synthesis Technique with Efficient Algorithms for Nearest Neighbor Classification. – Pattern Recognition Letters, Vol. **27**, 2006, pp. 1714-1724.
27. W a n g, S., X. Y a o. Multiclass Imbalance Problems: Analysis and Potential Solutions. – IEEE Trans. Syst., Man, Cybern. B, Vol. **42**, 2012, No 4, pp. 1119-1130.
28. L i, Y., X. Z h a n g. Improving k-Nearest Neighbour with Exemplar Generalization for Imbalanced Classification. Advances in Knowledge Discovery and Data Mining. – In: Lecture Notes in Computer Science. Vol. **6635**. 2011, pp. 321-332.
29. Z h a n g, J., I. M a n i. KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. – In: Proc. of International Conf. Machine Learning (ICML'2003), Workshop Learning from Imbalanced Data Sets, 2003.
30. L ó p e z, V., A. F e r n á n d e z, S. G a r c í a, V. P a l a d e, F. H e r r e r a. An Insight into Classification with Imbalanced Data: Empirical Results and Current Trends on Using Data Intrinsic Characteristics. – In: Information Sciences. Vol. **250**. Online Publication Date: 1 November 2013, pp. 113-141.
31. B o r s o s, Z., C. L e m n a r u, R. P o t o l e a. Dealing with Overlap and Imbalance: A New Metric and Approach. – Pattern Analysis and Applications. Online Publication Date: 27 September 2016.
32. Z h a n g, Z., B. K r a w c z y k, S. G a r c ì a, A. R o s a l e s-P é r e z, F. H e r r e r a. Empowering One-vs-One Decomposition with Ensemble Learning for Multi-Class Imbalanced Data. – Knowledge-Based Systems, Vol. **106**, 2016, pp. 251-263.
33. Z h o u, Z.-H., X.-Y. L i u. On Multi-Class Cost-Sensitive Learning. – Comput. Intell., Vol. **26**, 2010, No 3, pp. 232-257.