International Journal of Computational Intelligence Systems, Vol. 8, No. 4 (2015) 761-778

An effective Weighted Multi-class Least Squares Twin Support Vector Machine for Imbalanced data classification

Divya Tomar
Indian Institute of Information Technology
Allahabad, Uttar Pradesh, India
divyatomar26@gmail.com

Sonali Agarwal
Indian Institute of Information Technology
Allahabad, Uttar Pradesh, India
sonali@iiita.ac.in

Received 5 August 2014; Accepted 9 May 2015

Abstract

The performance of machine learning algorithms is affected by the imbalanced distribution of data among classes. This issue is crucial in various practical problem domains, for example, in medical diagnosis, network intrusion, fraud detection etc. Most efforts so far are mainly focused upon binary class imbalance problem. However, the class imbalance problem is also reported in multi-class scenario**.** The solutions proposed by the researchers for two-class scenario are not applicable to multi-class domains. So, in this paper, we have developed an effective Weighted Multi-class Least Squares Twin Support Vector Machine (WMLSTSVM) approach to address the problem of imbalanced data classification for multi class. This research work employs appropriate weight setting in loss function, e.g. it adjusts the cost of error for imbalanced data in order to control the sensitivity of the classifier. In order to prove the validity of the proposed approach, the experiment has been performed on fifteen benchmark datasets. The performance of proposed WMLSTSVM is analyzed and compared with some other SVMs and TWSVMs and it is observed that our proposed approach outperforms all of them. The proposed approach is statistically analyzed by using non-parametric Wilcoxon signed rank and Friedman tests.

Keywords: Least Squares Twin Support Vector Machine, Multi Least Squares Twin Support Vector Machine, Weighted Multi Least Squares Twin Support Vector Machine, Imbalanced data classification.

1. Introduction

Classification is one of the significant techniques of data mining which predicts the class label for any unknown input data. During the construction of a classifier, a learning algorithm identifies the relationship between class labels and attributes set. The performance of a classifier can deteriorate due to imbalanced distribution of data between classes in which each class contains different number of data points [1-5]. The imbalanced data problem occurs when one class includes large number of data points (referred as majority class) while other class includes less (referred as minority class). Standard classification approaches consider a balanced training dataset which induces a bias in favor of majority classes [6]. The degree of imbalance differs from one application domain to another and the correct class prediction of data points in an unusual class becomes more significant than the contrary case, for example, in disease diagnostic problem where the cases of diseases are unusual as compared to the normal population. So in this case, the correct recognition of a person with disease becomes more important. Therefore, a classifier could be beneficial if its prediction rate is higher for the disease category. In recent years, the class imbalance problem has attracted

significant interest due to its occurrence in application domains such as in identifying fraudulent telephone calls, disease diagnostic, intrusion detection, risk management, oil spills detection from radar images, text classification etc.[4-5,7-10].

Mostly, the solution provided by the researchers for the class imbalance problem has concentrated on two-class imbalance problem [1-2, 11-12]. However, the imbalance data distribution problem does not only exist in two-class. Multi-class data also suffers from this problem. It is more difficult to identify the majority and minority class for multi-class data. For example, one particular class X can be majority with respect to class Y, but minority with respect to another class Z. The solutions proposed for two-class imbalanced problems have been shown to be less effective for multiple class imbalanced learning. Some methods may not be directly applicable or may achieve poor performance [13-14]. For example, the solutions proposed by the researchers at data level are affected by the increased search space and also the solutions at algorithm level do not perform well and thus become more complicated for multi-class scenario [13-18]. So, there is a need to develop a classifier which effectively handles the imbalanced distribution of data in multi-classes. Therefore, in this research, we have proposed a classifier, termed as Weighted Multi-class Least Squares Twin Support Vector Machine (WMLSTSVM), for the multi class imbalance problem. In this classifier, we have assigned different weights to different data points. Initially, we have extended the formulation of binary Least Squares Twin Support Vector Machine (LSTSVM) to Multi Class Least Squares Twin Support Vector Machine (MLSTSVM) by using the concept of One-against-All (OAA). In the literature, it is found that One-against-One (OAO) concept based approach gives more accurate result in the presence of imbalanced data [14]. However, in this study we have adopted OAA concept to extend binary LSTSVM classifier due to the following reasons:

- For M-class classification problem, OAO MLSTSVM classifier constructs M (M-1) hyper-planes where (M-1) planes are for each class. While OAA MLSTSVM classifier generates M hyper-planes, one plane for each class. So, extending LSTSVM to multi-class by using OAO concept is a complex process.
- OAO MLSTSVM classifier takes more time in training as compared to the OAA MLSTSVM classifier.
- In, OAO MLSTSVM approach, each classifier is involved with the training data points of two classes at the same time. Therefore, the information of the remaining data points is omitted in each binary classification.

Next, weights are added to the formulation of proposed classifier so that it works well for both balanced and imbalanced distributed data involving multi-classes. Weights are introduced to control the sensitivity for imbalance ratio in determining each hyper-plane. The performance of the proposed approach has been compared with Multi-SVM, MBSVM, Adaboost.NC, OVO MLSTSVM and OVA-MLSTSVM on fifteen benchmark imbalanced dataset by using Geometric Mean metric. Some, pre-processing approaches such as static SMOTE, Global-CS and Random-Oversampling are combined with the base classifiers Multi-SVM and MBSVM to overcome the imbalance problem.

In this study, we have performed the statistical analysis of classifiers by following the recommendation of Demsar and make the statistical inferences from the observed difference in Geometric Mean [19]. Non-parametric Wilcoxon signed rank and Friedman's average rank hypothesis tests are used to statistically analyze the performance of the proposed classifier with existing classifiers. Results of Friedman's test are displayed with the help of modified version of Demsar significance diagram**.**

The paper is organized into eight sections. Section 2 and section 3 provide brief overview of imbalanced data problem in binary and multi-class scenario and their solutions. Section 4 and section 5 include background work and the proposed OAA MLSTSVM classifier respectively. The formulation of the proposed Weighted MLSTSVM approach to imbalance data distribution problems of multi-classes is discussed in section 6. Experiments on imbalanced datasets have been analyzed in section 7 and concluding remarks are given in section 8.

2. Imbalanced Data problem

In the classification field, the imbalanced data problem appears when data points belonging to each class vary

in numbers. The imbalanced data problem deteriorates the performance of classifiers as they consider the balanced distribution of data among classes. Various solutions have been suggested by the researchers to handle the imbalanced data problem. These solutions can be divided into three broad categories:

2.1. *Data level solutions*

Balancing distribution of data between classes is one of the simplest approaches. The objective of data level solutions is to rebalance the distribution of data between classes to reduce the effect of class imbalance [6, 12, 15, 20-27]. The benefit of using data level solutions is that they are independent of the selected classifier and hence are more versatile in nature. Since data level solutions are provided at the pre-processing time so we need to prepare the data only one time. Data level solutions can be divided into two groups: under-sampling and over-sampling methods. In under sampling method, the data points of prevalent class are reduced in such a way that each class contains equal number of data points. On the other hand, over sampling increases the data points of minority class to balance them with majority class. Sometimes the combination of both under sampling and over sampling approach is used to balance the distribution of data [25-28]. Under-sampling is a simple approach, but loses some significant information about prevalent class while over-sampling method generates an unnatural bias in favor of a minority class. Apart from this, balancing the data distributions by using these approaches also suffers from extra learning cost for analyzing and processing data [5]. Several under-sampling and over-sampling approaches are discussed below:

a. *Synthetic Minority Over-sampling (SMOTE):* It is one of the famous techniques of balancing the data through sampling [20]. Cieslak and Chawla have suggested SMOTE which generates synthetic instances along the line segments joining nearest neighbors of minority classes. Depending upon the amount of over-sampling required, neighbors are selected randomly from the k-nearest neighbors. SMOTE suffers from the over-generalization and variance problems.

b. *Random Over-sampling and Under-sampling:* The aim of random over-sampling method is to rebalance the distribution of data points between classes through random replication of the data points of minority class [15]. This method suffers from the over-fitting problem as it generates exact copies of existing data points. On the other hand, random under-sampling rebalances the distribution of data points between classes through random elimination of the data points of majority classes [15]. As this method remove the data points so it can discard potentially useful data which could be important for further processing.

c. *Informed Under*-sampling: It includes two approaches-EasyEnsemble and BalanceCascade [29]. The objective of these two approaches is to overcome the deficiency of information loss introduced in the conventional random under-sampling method [17, 29]. EasyEnsemble develops an ensemble learning system by independently sampling several subsets from the majority class. It then generates multiple classifiers based on the combination of each subset with the minority class data. This method can be considered as an unsupervised learning approach which explores the data of majority class by using independent random sampling with replacement. In contrast, Balance Cascade is a supervised learning approach that generates an ensemble of classifiers to systematically select majority class data points for under-sampling [29].

d. *Safe level SMOTE (SL-SMOTE):* As discussed earlier, SMOTE generates synthetic instances along the line segments joining nearest neighbors of minority classes, ignoring nearby data points of majority class. On the other hand, SL-SMOTE samples the data points of minority class along the same line with a different weight degree, called a safe level [30]. Safe level is obtained by using the k-nearest data points of minority class. The data point is considered to be safe if its safe value is close to 'k' while it is considered as a noise data point if the corresponding safe value is close to 0. Therefore, the aim of this approach is to generate synthetic data points in safe areas of the training set.

e. *Tomek Links:* It is used to remove the over-lapping that is introduced from sampling methods. Tomek Links can be defined as a pair of minimal distance nearest neighbor of opposite classes [31]. Given two data points x_i and x_j of different classes and $d(x_i, x_j)$ is the distance between them. Then, the pair (x_i, x_j) is called a Tomek Links, if there is no data point x_k, such that $d(x_i, x_k) < d(x_i, x_j)$ or $d(x_j, x_k) < d(x_i, x_j)$. Like this, if two data points form a Tomek Link, then either one of these data points is noise or both data points are border-line. This method can be used as an under-sampling method or as a data cleaning method. As an under-sampling method, Tomek Links eliminate the data points of majority class while as a data cleaning method it removes the data points of both classes.

f. *SMOTE + Edited Nearest Neighbor (SMOTE +ENN):* This method is also used to remove the data points from both classes [15]. ENN is applied after the SMOTE in order to eliminate any data point misclassified by its three nearest neighbors from the training dataset.

g. *One-sided Selection (OSS):* This is an under-sampling approach and is obtained by combining Tomek Links with Condensed Nearest Neighbor (CNN) rule. As discussed earlier, TL removes the noisy and borderline majority data points. CNN removes data points from the majority class that are distant from the decision border [32]. The remainder data points of majority class and all data points of minority class are used for learning.

h. Cluster based Sampling Method: Jo and Japkowicz proposed Cluster based Oversampling (CBO) approach to handle the within-class imbalance problem [33]. In another research work, Yen and Lee also proposed cluster based under-sampling approach to deal with the imbalanced data problem [34]. Cluster based under-sampling approach improves the predictive accuracy of minority class by selecting the representative data as training data. They also investigated the effect of under-sampling approach in the imbalanced class distribution scenario. To handle imbalanced data problem, Chen et al. proposed a novel over-sampling method based on cluster ensemble [35]. This approach first generates multiple partitions by using cluster ensembles and matches these clusters with different partitions. Then, it searches for cluster boundary minority data points with the help of clustering consistency index and finally the minority data points are over-sampled around the border between clusters.

2.2. *Algorithmic level solutions*

The second solution of the class imbalance problem is adjusting the classifier which is an efficient approach and provides better results as compared to the previous one. These solutions can be defined as internal methods that develop new algorithms or modify existing ones in order to solve the class imbalance problem [36-40]. One-class learning is useful approach suitable for imbalanced datasets with high-dimensional noisy features [41]. In this approach, a classifier learns to predict the data points of one class which is usually minority class. The main focus of this approach is to separate the data points of minority class from majority class. One-class learning adopts two strategies-The first strategy identifies the data points of the target class (usually a minority class) instead of discriminating the data points of all classes. While the second strategy considers the data points of both classes and uses internal bias strategies to predict the target class [41-42]. REMED (Rule Extraction for MEdical Diagnosis) and RIPPER are two examples of one-class learning approach [41]. Classifier ensembles are one of the important approaches used by the researchers to handle the class imbalance problem. Classifier ensembles learn from a set of classifiers rather than one classifier and predict the class of a new data point by combining the predictions of all classifiers used for ensemble. Boosting and random forest are two commonly used approaches to ensemble classifiers [43-44].

2.3. *Cost –sensitive solutions*

Sampling methods focus on balancing the distributions of data points between classes while cost-sensitive learning methods take into account the costs associated with misclassifying data points [45-51]. It considers the variable cost of a misclassification of the different classes. The cost-sensitive learning approaches try to minimize the total misclassifications cost, but minority class gains importance in this cost function. Cost sensitive learning methods solve the data imbalance problem by using different cost matrices that describe the costs of classifying data points from one class to another. It is found from the literature that cost-sensitive learning based solutions are more effective than sampling methods [52-54].

a. *Cost sensitive learning framework:* It is based on the concept of cost matrix which can be considered as a numerical representation of the penalty of misclassifying data points [17]. Let us consider a binary classification scenario, the cost of misclassifying a data point of majority class into minority class is $C(min, maj)$ and the cost of opposite case is $C(maj, min)$. Usually, there is no cost for correct classification of data points of either class and $C(maj, min) > C(min, maj)$. The objective of cost-sensitive learning is to minimize the overall cost on the training dataset and these concepts are easily extended to multi-class classification scenario by considering $C(i, j)$, which indicates the cost of misclassifying the data point of j^{th} class into i^{th} class.

b. Cost-sensitive Decision Trees: In this approach, the cost-sensitive fitting can take three forms. First, one can apply cost to the decision threshold. Maloof used a

decision tree threshold moving approach for the classification of imbalanced data with unknown or unequal misclassification costs [55]. In another research work, Breiman et al. established the relationship between misclassification costs of each class, the distribution of training data points, and the placement of decision tree threshold [56]. Second, cost can be given to the split criteria at each node. The main task in this is to fit an impurity function which is insensitive to unequal costs. Usually, accuracy is used as the impurity function for decision trees, which selects the split with minimal error at each node. However, accuracy is sensitive to imbalanced distributed data points. Drummond and Holte have used three specific impurity metrics, Gini, DKM and entropy and obtained improved cost-sensitivity as compared to the accuracy/error rate [57]. Finally, cost-sensitive decision tree schemes can be applied for pruning. In decision tree, pruning is beneficial because it improves the generalization by removing leaves with class probability estimates below a specified threshold. Elkan have used Laplace smoothing method of the probability estimate and Laplace pruning technique [45].

c. *Cost-sensitive Neural Network:* Cost sensitivity can be introduced to Neural Networks in four ways: first, one can apply cost sensitive modifications to the probabilistic estimate, second, the output of Neural Network can be made cost-sensitive, third, cost can be applied to the error minimization function, and lastly cost-sensitive modification can be applied to the learning rate. Kukar and Kononenko proposed cost sensitive Neural Network and applied cost in the testing phase to modify the probability estimate of the output [58]. They also applied cost-sensitive modification to the output of Neural Network. They have modified the output during training phase to bias the Neural Network to focus more on the rare class.

In another research work, Sun et al. proposed three cost-sensitive boosting methods, AdaC1, AdaC2, and AdaC3 for handling imbalanced learning. They have added cost into the weight updating strategy of AdaBoost [5]. AdaCost is another cost-sensitive boosting algorithm in which cost sensitivity is applied inside the exponent of the weight updating formula of AdaBoost [59]. It uses a cost-adjustment function which decreases the weights of correctly classified data points and increases the weights of costly misclassification. Cost functions have also been integrated with Support Vector Machine and Bayesian classifiers [60-66].

3. Multi-class Imbalance problems

This section discusses the different methodologies proposed by the researchers to solve multi-class imbalanced problems as:

3.1. *Static SMOTE*

This is the pre-processing approach in which the resampling procedure is applied in M steps, where M is the number of classes [67]. In each iteration, this procedure chooses the class of minimum size and replicates the number of data points of the class in the original data-set. Synthetic data points are generated by applying the SMOTE algorithm only over the data points of the minority class. SMOTE duplicates the minority class by taking into account only the data points of original dataset.

3.2. *Global-CS*

Zhou and Liu resampled each class to equilibrate the significance of the data points of different classes in imbalanced problem scenario [68]. They replicated each data point of class i $\left\lfloor {N_{max}}/{N_i} \right\rfloor$ times and selected $N_{max}\%N_i$ additional random data points from the dataset, where N_i and N_{max} denote number of data points of the ith class and majority class respectively.

3.3. *AdaBoost.NC*

Wang and Yao have analyzed the impact of multi-minority class and multi-majority class on the performance of random under-sampling and over-sampling methods and proposed AdaBoost.NC to handle the imbalance problem in multi-class scenario [13]. AdaBoost.NC is an ensemble learning approach which combines the strength of negative correlation learning and boosting. In this approach, the weights of the data points are updated with an ad hoc formula which is based on the classification or misclassification given by both the classifier learned in the current iteration and the global ensemble.

3.4. *Other methods*

Most of the solutions given by the researchers to handle class imbalance problem in multi-class scenario use decomposition schemes and work with binary class imbalance solutions. Tan et al. have used both OAA and OAO approach to break down the protein-fold classification problem and then developed rule-based learners to enhance the coverage of data points of minority class [69]. Zhao et al. also used OAA and SMOTE approach to handle the issue of class imbalance in multi-class scenario [70]. Chen et al. proposed OAA

based algorithm to decompose the multi-class problem into binary problems and applied some advanced sampling approaches to rebalance the distribution of data [71]. Liao et al. analyzed several over-sampling and under-sampling approach with OAA for the classification of Weld flaws with imbalanced data [72]. In another research work, Fernandez have combined OAO and SMOTE sampling approach to handle the multi-class imbalance problem [73]. MetaCost is another solutions to multi class problems which make a cost sensitive classifier [74]. Different from the decomposition schemes to handle the imbalanced learning in multi-class scenario, Sun et al. have proposed a cost sensitive boosting algorithm to enhance the classifier performance for the multi class imbalance problem [75]. The main focus of [75] was to search for the cost matrix and for this purpose Genetic Algorithm was used to find the optimum cost setup for each class.

4. Background Work

Recently, Twin Support Vector Machine (TWSVM) has attracted the attention of researchers due to its better performance and speed. TWSVM is a binary classifier introduced by Jayadeva et al. and classifies the data points of two classes by using a pair of non-parallel planes [76]. TWSVM is based on two well-known techniques -Support Vector Machine (SVM) and Generalized Eigen-value Proximal SVM (GEPSVM). SVM is a binary classifier which classifies the data points of two classes by constructing an optimal separating hyper-plane [77]. While, GEPSVM, proposed by Mangasarian et al., produces a pair of non-parallel hyper-planes for the separation of data points of two classes [78]. TWSVM solves two simple Quadratic Programming Problems (QPPs) in place of single complex QPP as in traditional SVM. In SVM, all data points together give constraints to QPP while in TWSVM, data points of one class give constraints to other QPP and vice versa. The validity and effectiveness of the TWSVM has been proved over conventional SVM and GEPSVM on several benchmark datasets in [76]. Later, Kumar et al. introduced a new binary classifier named as LSTSVM which is the least squares version of TWSVM [79]. LSTSVM classifies the data points of two classes by optimizing two linear equations with equality constraints as opposed to the TWSVM which solves two QPPs with inequality constraints. LSTSVM is a simple binary classifier and has shown better generalization ability as compared to TWSVM. This section presents the brief overview of traditional TWSVM and LSTSVM. Consider the training dataset 'T' contains 'n' data points and is represented as:

$$T = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \quad (1)$$

where $x_i \in R^d$ indicates i[th] data point and $y_i \in \{1, -1\}$ is the class label. Suppose, positive and negative class comprise n_1 and n_2 data points respectively and $n=n_1 + n_2$.

4.1. *Twin Support Vector Machine*

For training dataset 'T', TWSVM generates following decision function:

$$f(x) = arg\ \min_{i=1,2} \frac{|w_i.x+b_i|}{\|w_i\|} \quad (2)$$

by finding two non-parallel hyper-planes

$$x^T w_1 + b_1 = 0 \quad \text{and} \quad x^T w_2 + b_2 = 0 \quad (3)$$

where $w_1, w_2 \in R^d$ are normal vectors to the hyper-planes and $b_1, b_2 \in R$ are bias terms. TWSVM classifies the data points by optimizing following two QPPs:

$$\min(w_1, b_1, \xi) \quad \frac{1}{2}\|Aw_1 + e_1 b_1\|^2 + c_1 e_2^T \xi$$
$$\text{s.t.} \quad -(Bw_1 + e_2 b_1) + \xi \geq e_2, \xi \geq 0 \quad (4)$$
$$\min(w_2, b_2, \eta) \quad \frac{1}{2}\|Bw_2 + e_2 b_2\|^2 + c_2 e_1^T \eta$$
$$\text{s.t.} \quad (Aw_2 + e_1 b_2) + \eta \geq e_1, \eta \geq 0 \quad (5)$$

where matrices $A \in R^{n_1 \times d}$ and $B \in R^{n_2 \times d}$ comprise the data points of positive and negative class respectively. $e_1 \in R^{n_1}$ *and* $e_2 \in R^{n_2}$ are the vectors of 1's, c_1 , c_2 >0 are penalty parameters, and $\xi \in R^{n_2}$ and $\eta \in R^{n_1}$ are slack variables corresponding to negative and positive class.

4.2. *Least Squares Twin Support Vector Machine*

LSTSVM classifies the data points by optimizing following two linear equations rather than two QPPs with equality constraints:

$$\min(w_1, b_1, \xi) \quad \frac{1}{2}\|Aw_1 + e_1 b_1\|^2 + \frac{c_1}{2}\xi^T \xi$$
$$\text{s.t.} \quad -(Bw_1 + e_2 b_1) + \xi = e_2 \quad (6)$$

and

$$\min(w_2, b_2, \eta) \quad \frac{1}{2}\|Bw_2 + e_2 b_2\|^2 + \frac{c_2}{2}\eta^T \eta$$
$$\text{s.t.} \quad (Aw_2 + e_1 b_2) + \eta = e_1 \quad (7)$$

Solution of above two equations determines the values of normal vector and bias as:

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -\left(F^T F + \frac{1}{c_1} E^T E\right)^{-1} F^T e_2 \quad (8)$$

and

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = \left(E^T E + \frac{1}{c_2} F^T F\right)^{-1} E^T e_1 \quad (9)$$

where, E= [A e_1] and F=[B e_2]. Further, these values find two non-parallel hyper-planes according to

equation 3. A test data point is assigned to a class by using following decision function:

$$f(x) = arg\ \min_{i=1,2} \frac{|w_i.x+b_i|}{\|w_i\|} \quad (10)$$

where | . | indicates the perpendicular distance of test data point from hyper-plane. Kumar et al. also proposed the formulation of LSTSVM for non-linearly separable data points by using kernel trick. Non-linear LSTSVM constructs two kernel generated surfaces in higher-dimension as:

$$K(x^T, Z^T)\mu_1 + \gamma_1 = 0 \text{ and } K(x^T, Z^T)\mu_2 + \gamma_2 = 0 \quad (11)$$

where Z=[A B]^T and K is any arbitrary chosen kernel function. Following are the optimization problems of non-linear LSTSVM:

$$\min(\mu_1, \gamma_1, \xi) \quad \frac{1}{2}\|K(A, Z^T)\mu_1 + e_1\gamma_1\|^2 + \frac{c_1}{2}\xi^T\xi$$
$$\text{s.t.} \quad -(K(B, Z^T)\mu_1 + e_2\gamma_1) = e_2 - \xi \quad (12)$$

and

$$\min(\mu_2, \gamma_2, \eta) \quad \frac{1}{2}\|K(B, Z^T)\mu_2 + e_2\gamma_2\|^2 + \frac{c_2}{2}\eta^T\eta$$
$$\text{s.t.} \quad (K(A, Z^T)\mu_2 + e_1\gamma_2) = e_1 - \eta \quad (13)$$

The solution of above problem produces following values:

$$\begin{bmatrix}\mu_1 \\ \gamma_1\end{bmatrix} = -(H^T H + \frac{1}{c_1} G^T G)^{-1} H^T e_2 \quad (14)$$

$$\begin{bmatrix}\mu_2 \\ \gamma_2\end{bmatrix} = (G^T G + \frac{1}{c_2} H^T H)^{-1} G^T e_1 \quad (15)$$

where G= [K (A, Z^T) e_1] , H= [K (B, Z^T) e_2] and the class is assigned to test data point as:

$$\text{class(j)} = \text{argmin(j} = 1,2) \frac{|K(x^T, Z^T)\mu_j + \gamma_j|}{\|\mu_j\|} \quad (16)$$

5. Multiclass Least Squares Twin Support Vector Machine

Mostly, real life applications contain multiple classes and demand for a classifier that works effectively for the categorization of multiple classes. As discussed earlier, LSTSVM has shown better generalization performance, but it is suitable only for two-class problem. However, its multi-class extension is rarely noted in the literature. So, in this paper, we have proposed a novel classifier named as MLSTSVM which is the multi class extension of the binary LSTSVM classifier. The proposed classifier works on "One-against-All" strategy in which the data points of each class is trained with the data points of other classes. For M-class, MLSTSVM constructs M-binary LSTSVM classifiers and ith classifier (where i=1,2,…,M) considers the data points of ith class as positive data points and data points of other classes as negative data points. In this manner, it solves M-linear programming equations and seeks M-hyper planes, one for each class.

Let the training dataset includes 'n' data points: $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, where $x_i \in R^d$ indicates feature vector and $y_i \in \{1,2, \dots, M\}$ indicates corresponding class label. Let the matrix $A_i \in R^{n_i \times d}$ represents the data points of ith class, where n_i denotes the size of ith class. Again, consider the matrix $B_i \in R^{(n-n_i)\times d}$ is comprised of the data points of all classes except ith class and is defined as:

$$B_i = [(A_1)^T, (A_2)^T, \dots, (A_{i-1})^T, (A_{i+1})^T, \dots, (A_M)^T]^T \quad (17)$$

Here, we present the formulation of MLSTSVM for both linear and non-linear separable data points as:

5.1. *Linear MLSTSVM*

Taking the ith class as an example, let the ith hyper-plane is:

$$(w_i.x)+b_i=0 \quad \text{, where i=1,2,…,M} \quad (18)$$

where $w_i \in R^d$ and b_i indicate normal vector to the hyper-plane and bias term respectively. The objective function of ith classifier is obtained as:

$$\min(w_i, b_i, \xi_i) \quad \frac{1}{2}\|A_i w_i + e_{i1} b_i\|^2 + \frac{c_i}{2}\xi_i^T\xi_i$$
$$\text{s.t.} \quad (B_i w_i + e_{i2} b_i) + \xi_i = e_{i2} \quad (19)$$

where c_i>0 is the penalty parameter, $e_{i1} \in R^{n_i}$ and $e_{i2} \in R^{(n-n_i)}$ are the vector of 1's and ξ_i is the slack variable. The first term of the objective function minimizes the squared sum distance of the data points of ith class from the hyper plane and tries to keep the hyper-plane in its close affinity. The second term minimizes the sum of misclassification error due to data points belonging to rest of the M-1 classes. Thus the minimization of the above objective function keeps the ith hyper-plane near to the data points of ith class and far from the data points of rest of the classes. Lagrangian corresponding to the equation 19 is achieved as:

$$L(w_i, b_i, \xi_i, \alpha_i) = \frac{1}{2}\|A_i w_i + e_{i1} b_i\|^2 + \frac{c_i}{2}\xi_i^T\xi_i - \alpha_i^T((B_i w_i + e_{i2} b_i) + \xi_i - e_{i2}) \quad (20)$$

where α_i is a non-negative lagrangian multiplier. Following necessary Karush-Kuhn-Tucker (KKT) conditions are obtained by differentiating equation 20 with respect to w_i, b_i, ξ_i *and* α_i :

$$\frac{\partial L}{\partial w_i} = A_i^T(A_i w_i + e_{i1} b_i) - B_i^T\alpha_i = 0 \quad (21)$$

$$\frac{\partial L}{\partial b_i} = e_{i1}^T(A_i w_i + e_{i1} b_i) - e_{i2}^T\alpha_i = 0 \quad (22)$$

$$\frac{\partial L}{\partial \xi_i} = c_i\xi_i - \alpha_i = 0 \quad (23)$$

$$\frac{\partial L}{\partial \alpha_i} = (B_i w_i + e_{i2} b_i) + \xi_i - e_{i2} = 0 \quad (24)$$

Equations 21 and 22 lead to:

$$\begin{bmatrix} A_i^T \\ e_{i1}^T \end{bmatrix} [A_i \; e_{i1}] \begin{bmatrix} w_i \\ b_i \end{bmatrix} - \begin{bmatrix} B_i^T \\ e_{i2}^T \end{bmatrix} \alpha_i = 0 \quad (25)$$

Define $E_i = [A_i \; e_{i1}]$, $F_i = [B_i \; e_{i2}]$ and $u_i = \begin{bmatrix} w_i \\ b_i \end{bmatrix}$. With these notations equation 25 may be reformulated as:

$$E_i^T E_i u_i - F_i^T \alpha_i = 0 \quad (26)$$

$$u_i = (E_i^T E_i)^{-1} F_i^T \alpha_i \quad (27)$$

It is observed that the objective function of proposed classifier requires the inverse of $E_i^T E_i$ which is sometime difficult to calculate due to ill-conditioned. To avoid the possibility of ill-conditioning of matrix, a regularization term εI is added to the above formulation as:

$$u_i = (E_i^T E_i + \varepsilon I)^{-1} F_i^T \alpha_i \quad (28)$$

where $\varepsilon > 0$ is a very small scalar and I is an identity matrix of suitable size. The value of lagrangian multiplier is computed from (23) and (24) as:

$$\alpha_i = c_i(e_{i2} - F_i u_i) \quad (29)$$

Substituting (27) with (29):

$$\begin{bmatrix} w_i \\ b_i \end{bmatrix} = u_i = (F_i^T F_i + \frac{1}{c_i} E_i^T E_i)^{-1} F_i^T e_{i2} \quad (30)$$

The above values generate the hyper-plane for i^{th} classifier. In the same way, a hyper-plane is generated for each classifier and a class is assigned to new data point depending on which plane lies nearest to it. The decision function is represented as:

$$f(x) = \arg\min_{i=1,..,M} \frac{|w_i.x + b_i|}{\|w_i\|} \quad (31)$$

Figure 1 represents the geometric representation of linear MLSTSVM for three classes. Different shapes represent the data points of different classes. Figure shows three hyper-planes, plane 1, plane 2 and plane 3 for class 1, class 2 and class 3 correspondingly in such a way that data points of each class lie in the close proximity of the corresponding hyper-plane while as far as possible from other hyper-planes.

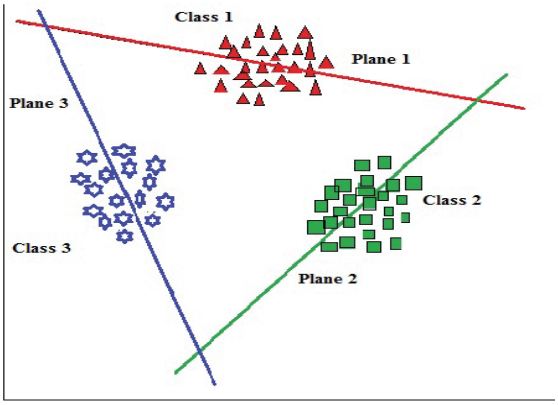

Fig. 1. Geometric representation of Linear MLSTSVM

5.2. *Non-Linear MLSTSVM*

Sometime, it is not possible to separate the data points with linear class boundaries. So, we have extended the formulation of proposed MLSTSVM to non-linear cases by using kernel trick. Firstly, kernel function is used to transform the data points in higher dimensional space and then MLSTSVM classifier constructs kernel surfaces in that space. Equation of i^{th} kernel surfaces is obtained as:

$$K(x, Z^T)\mu_i + \gamma_i = 0 \quad \text{where i=1,..., M} \quad (32)$$

where $Z=[A_i \; B_i]^T$ and 'K' is appropriately chosen kernel function. The non-linear MLSTSVM classifier is constructed by solving following optimization problem:

$$\min(\mu_i, \gamma_i, \xi_i) \quad \frac{1}{2}\|K(A_i, Z^T)\mu_i + e_{i1}\gamma_i\|^2 + \frac{c_i}{2}\xi_i^T\xi_i$$

$$\text{s.t.} \quad (K(B_i, Z^T)\mu_i + e_{i2}\gamma_i) + \xi_i = e_{i2} \quad (33)$$

Lagrangian function corresponding to (33) is given by:

$$L(\mu_i, \gamma_i, \xi_i, \alpha_i) = \frac{1}{2}\|K(A_i, Z^T)\mu_i + e_{i1}\gamma_i\|^2 + \frac{c_i}{2}\xi_i^T\xi_i - \alpha_i^T((K(B_i, Z^T)\mu_i + e_{i2}\gamma_i) + \xi_i - e_{i2}) \quad (34)$$

KKT conditions for (34) are:

$$\frac{\partial L}{\partial \mu_i} = K(A_i, Z^T)^T(K(A_i, Z^T)\mu_i + e_{i1}\gamma_i) - K(B_i, Z^T)^T\alpha_i = 0 \quad (35)$$

$$\frac{\partial L}{\partial \gamma_i} = e_{i1}^T(K(A_i, Z^T)\mu_i + e_{i1}\gamma_i) - e_{i2}^T\alpha_i = 0 \quad (36)$$

$$\frac{\partial L}{\partial \xi_i} = c_i\xi_i - \alpha_i = 0 \quad (37)$$

$$\frac{\partial L}{\partial \alpha_i} = (K(B_i, Z^T)\mu_i + e_{i2}\gamma_i) + \xi_i - e_{i2}) = 0 \quad (38)$$

Equations (37) and (38) lead to:

$$\begin{bmatrix} K(A_i, Z^T)^T \\ e_{i1}^T \end{bmatrix} [K(A_i, Z^T) \; e_{i1}] \begin{bmatrix} \mu_i \\ \gamma_i \end{bmatrix} - \begin{bmatrix} K(B_i, Z^T)^T \\ e_{i2}^T \end{bmatrix} \alpha_i = 0 \quad (39)$$

Define $G_i = [K(A_i, Z^T) \; e_{i1}]$ and $H_i = [K(B_i, Z^T) \; e_{i2}]$. Using these notations, equation (39) may be rewritten as:

$$G_i^T G_i \begin{bmatrix} \mu_i \\ \gamma_i \end{bmatrix} - H_i^T \alpha_i = 0 \quad (40)$$

Normal vector and bias are obtained from (37), (38) and (40) as:

$$\begin{bmatrix} \mu_i \\ \gamma_i \end{bmatrix} = (H_i^T H_i + \frac{1}{c_i} G_i^T G_i)^{-1} H_i^T e_{i2} \quad (41)$$

The decision function for non-linear MLSTSVM is represented as:

$$f(x) = arg\min_{m=1,..,M} \frac{|\mu_m.K(x, D^T) + \gamma_m|}{\|\mu_m\|} \quad (42)$$

A class is assigned to new data point depending on which kernel surface lies nearest to it.

6. Weighted Multi-class Least Squares Twin Support Vector Machine

Earlier, we have discussed several approaches to handle class imbalance problem and it is observed that mostly two-class imbalance problem is targeted by the researchers. So, in this paper we have focused on multi-class imbalance problem and proposed a novel Weighted MLSTSVM classifier (WMLSTSVM). In this approach, different weights are assigned to the data points of different classes. Therefore, the selection of appropriate weight is an important issue of consideration. The proposed approach selects and assigns a weight to the classes according to their size. Consider $n_1, n_2,...,n_M$ be the size of classes, where M is the number of classes present in the dataset and $n=n_1 + n_2 +...+ n_M$. The weight is assigned to a class according to the following formula:

$$p_i = \frac{n-n_i}{(M-1)n}, \quad \text{where i=1,2,...,M} \quad (43)$$

We can draw following three conclusions from the above mentioned formula:

i. Higher weight is assigned to the class with small data points while lower weight is assigned to the class with large data points so that each class could get equal importance during the training of classifier.
ii. $p_i \in (0,1)$, so that the proposed classifier could be trained with convergence.
iii. The weights are normalized without loss of generality and $\sum_{i=1}^{m} p_i = 1$.

The formulations of the proposed WMLSTSVM for linear and non-linear cases are obtained as:

6.1. *Linear WMLSTSVM*

The objective function of linear WMLSTSVM for i^{th} class is formulated as:

$$\min(w_i, b_i, \xi_i) \quad \frac{1}{2}\|A_i w_i + e_{i1} b_i\|^2 + \frac{c_i}{2}\xi_i^T W_i \xi_i$$
$$\text{s.t.} \quad (B_i w_i + e_{i2} b_i) + \xi_i = e_{i2} \quad (44)$$

where $W_i \in R^{(n-n_i)\times(n-n_i)}$ represents the diagonal matrix containing weights for the data points of i^{th} class as per equation 43. The lagrangian function of the above mentioned objective function is achieved as:

$$L(w_i, b_i, \xi_i, \alpha_i) = \frac{1}{2}\|A_i w_i + e_{i1} b_i\|^2 + \frac{c_i}{2}\xi_i^T W_i \xi_i - \alpha_i^T((B_i w_i + e_{i2} b_i) + \xi_i - e_{i2}) \quad (45)$$

where α_i is a non-negative lagrangian multiplier. Necessary KKT optimality conditions for above objective function are obtained as:

$$\frac{\partial L}{\partial w_i} = A_i^T(A_i w_i + e_{i1} b_i) - B_i^T \alpha_i = 0 \quad (46)$$

$$\frac{\partial L}{\partial b_i} = e_{i1}^T(A_i w_i + e_{i1} b_i) - e_{i2}^T \alpha_i = 0 \quad (47)$$

$$\frac{\partial L}{\partial \xi_i} = c_i W_i \xi_i - \alpha_i = 0 \quad (48)$$

$$\frac{\partial L}{\partial \alpha_i} = (B_i w_i + e_{i2} b_i) + \xi_i - e_{i2} = 0 \quad (49)$$

Equations (46) and (47) determine:

$$\begin{bmatrix} A_i^T \\ e_{i1}^T \end{bmatrix} [A_i \; e_{i1}] \begin{bmatrix} w_i \\ b_i \end{bmatrix} - \begin{bmatrix} B_i^T \\ e_{i2}^T \end{bmatrix} \alpha_i = 0 \quad (50)$$

Define $E_i = [A_i \; e_{i1}]$, $F_i = [B_i \; e_{i2}]$ and $u_i = \begin{bmatrix} w_i \\ b_i \end{bmatrix}$. With these notations equation 50 may be rewritten as:

$$E_i^T E_i u_i - F_i^T \alpha_i = 0 \quad (51)$$

$$u_i = (E_i^T E_i)^{-1} F_i^T \alpha_i \quad (52)$$

Lagrangian multiplier is determined from (48), (49) and (51) as:

$$\alpha_i = \left[\frac{W_i^{-1}}{c_i} + F_i(E_i^T E_i)^{-1} F_i^T\right]^{-1} e_{i2} \quad (53)$$

w_i and b_i are obtained from (52) and (53) and further seek non-parallel hyper-plane according to (18). So, the difference between MLSTSVM and WMLSTSVM is that W_i^{-1} is an extra term in the lagrangian multiplier. Decision function of WMLSTSVM is same as of MLSTSVM. Due to lagrangian multiplier, the value of normal vectors and biases differ and so hyper-planes. But the decision regarding class assignment is same as in MLSTSVM. For each new data point, its perpendicular distance is measured from each hyper-plane and the data point is assigned to the class closest to it as given in (31).

Algorithm:

For Training:

For i=1 to M, where M is number of classes in dataset.
1a. Define weight for each class according to (43).
1b. Obtain matrices E_i and F_i as:

$$E_i=[A_i \; e_{i1}] \text{ and } F_i=[B_i \; e_{i2}]$$

where A_i includes the data point of i^{th} class and B_i includes the data points of rest of the classes and defined by (17).
1c. Penalty parameters are selected on the basis of validation.
1d. Normal vector and bias are determined from (52) and (53) and generate hyper-plane using (18).

For Testing:

Training phase generates M hyper-planes one for each class. During testing phase, the distance of a test data point is calculated from each hyper-plane and a class, corresponding to the hyper-plane which is located at minimum distance from test data point, is assigned to it. The decision function regarding class assignment is mentioned in (31).

6.2. *Non-Linear WMLSTSVM*

WMLSTSVM is efficiently extended to non-linear cases by utilizing kernel trick. It also generates kernel

surfaces same as in MLSTSVM to separate the data points. The formulation of non-linear WMLSTSVM is obtained as:

$$\min(\mu_i, \gamma_i, \xi_i) \quad \frac{1}{2} \|K(A_i, Z^T)\mu_i + e_{i1}\gamma_i\|^2 + \frac{c_i}{2} \xi_i^T W_i \xi_i$$
$$\text{s.t.} \quad (K(B_i, Z^T)\mu_i + e_{i2}\gamma_i) + \xi_i = e_{i2} \qquad (54)$$

Lagrangian function corresponding to (54) is formulated as:

$$L(\mu_i, \gamma_i, \xi_i, \alpha_i) = \frac{1}{2} \|K(A_i, Z^T)\mu_i + e_{i1}\gamma_i\|^2 + \frac{c_i}{2} \xi_i^T W_i \xi_i - \alpha_i^T((K(B_i, Z^T)\mu_i + e_{i2}\gamma_i) + \xi_i - e_{i2}) \qquad (55)$$

KKT conditions for (55) are given below:

$$\frac{\partial L}{\partial \mu_i} = K(A_I, Z^T)^T(K(A_I, Z^T)\mu_i + e_{i1}\gamma_i) - K(B_I, Z^T)^T\alpha_i = 0 \qquad (56)$$

$$\frac{\partial L}{\partial \gamma_i} = e_{i1}^T(K(A_i, Z^T)\mu_i + e_{i1}\gamma_i) - e_{i2}^T\alpha_i = 0 \qquad (57)$$

$$\frac{\partial L}{\partial \xi_i} = c_i W_i \xi_i - \alpha_i = 0 \qquad (58)$$

$$\frac{\partial L}{\partial \alpha_i} = (K(B_i, Z^T)\mu_i + e_{i2}\gamma_i) + \xi_i - e_{i2}) = 0 \qquad (59)$$

Following equation is obtained by combining (56) and (58):

$$\begin{bmatrix} K(A_i, Z^T)^T \\ e_{i1}^T \end{bmatrix} [K(A_i, Z^T) \ e_{i1}] \begin{bmatrix} \mu_i \\ \gamma_i \end{bmatrix} - \begin{bmatrix} K(B_i, Z^T)^T \\ e_{i2}^T \end{bmatrix} \alpha_i = 0 \qquad (60)$$

Define $G_i = [K(A_i, Z^T) \ e_{i1}]$ and $H_i = [K(B_i, Z^T) \ e_{i2}]$. With these notations, (60) is reformulated as:

$$G_i^T G_i \begin{bmatrix} \mu_i \\ \gamma_i \end{bmatrix} - H_i^T \alpha_i = 0 \qquad (61)$$

$$\begin{bmatrix} \mu_i \\ \gamma_i \end{bmatrix} = (G_i^T G_i)^{-1} H_i^T \alpha_i \qquad (62)$$

Lagrangian Multiplier α_i is determined from (58), (59) and (61):

$$\alpha_i = \left[\frac{W_i^{-1}}{c_i} + H_i(G_i^T G_i)^{-1} H_i^T\right]^{-1} e_{i2} \qquad (63)$$

These values are used to construct kernel-generated surface as per equation 32. The class is allocated to test data point by using (42).

Algorithm:

For Training:

Choose Kernel Function.

For i=1 to M, where M is number of classes in dataset.

1a. Define weight for each class according to (43).

1b. Obtain matrices G_i and H_i as:

$G_i = [K(A_i, Z^T) \ e_{i1}]$ *and* $H_i = [K(B_i, Z^T) \ e_{i2}]$

where A_i includes the data point of i^{th} class and B_i includes the data points of rest of the classes and defined by (17).

1c. Penalty parameters are selected on the basis of validation.

1d. Normal vector and bias are determined from (62) and (63) and generate kernel surface using (32).

For Testing:

Training phase generates M kernel surfaces one for each class. During testing phase, a class is assigned to test data point by using (42).

7. Experiments and Discussion

7.1. *Dataset Description*

In order to prove the validity of the proposed methodology, we have performed experiment on fifteen multi-class imbalanced benchmark datasets taken from KEEL dataset repository [80]. Table 1 indicates the details of benchmark datasets used in this research work. The datasets contain multiple classes and the class with large size (number of data points) is considered as majority class and with small size is considered as minority class among all classes. Imbalance ratio is calculated by taking the ratio of the size of majority class with minority class.

Table 1. Details of benchmark datasets

Dataset	Data size	Features	Classes	Imbalance Ratio
Balance(Bal)	625	4	3	5.88
Ecoli(Eco)	336	7	8	71.5
Glass(Gls)	214	9	6	8.44
Wine(Win)	178	13	3	1.5
NewThyroid(Thy)	215	5	3	4.84
Hayes Roth(HaR)	132	4	3	1.7
Dermatology(Der)	366	34	6	5.55
Shuttle(Shu)	2175	9	5	853
Pen Based(PnB)	1100	16	10	1.95
PageBlock(PgB)	548	10	5	164
Contraceptive(Con)	1473	9	3	1.89
Lymphography(Lym)	148	18	4	40.5
Zoo	101	16	7	10.25
Splice(Spl)	3190	60	3	2.16
Cleveland(Cld)	467	13	5	12.62

7.2. *Performance Evaluation Measures*

The performance of proposed WMLSTSVM classifier is evaluated by using Geometric Mean which is calculated from confusion matrix. Here, we have determined the confusion matrix for multiple classes as indicated in table 2.

Table 2. Confusion Matrix

Predicted Class ↓	Actual Class →			
	$Class_1$	$Class_2$	...	$Class_M$
$Class_1$	C_{11}	C_{12}	...	C_{1M}
$Class_2$	C_{21}	C_{22}	...	C_{2M}
...	...	...	...	...
$Class_M$	C_{M1}	C_{M2}	...	C_{MM}

Count C_{ii} , also referred as True Prediction (TP), indicates the number of data points of class y_i which are correctly classified into it. While count C_{ij} of class y_i with respect to the class y_j ($y_i \neq y_j$) defines the number of data points of class y_j which are incorrectly classified into class y_i by the classifier. C_{ij} also referred as False prediction (FP) and for class y_i , $FP(i)=\sum_{j=1,j\neq i}^{M} c_{ij}$. Counts obtain from confusion matrix are used to determine the performance metrics such as True Positive Rate (TPR) or recall and Geometric Mean (G-Mean). TPR of class y_i is formulated as:

$$TPR_i = \frac{c_{ii}}{\sum_{j=1}^{M} c_{ij}} \tag{64}$$

If number of class M=2, then TPR_1 and TPR_2 also referred as sensitivity and specificity respectively. Geometric Mean (G-Mean) is a performance evaluation metric proposed by Kubat et al. for two class problems [4]. It measures the balanced performance of a classifier and is obtained by taking the geometric mean of recall values of two classes. For multiple classes, the G-Mean is calculated by taking the geometric means of recall values of every class as:

$$Geometric\ Mean = \sqrt[M]{\prod_{i=1}^{M} TPR_i} \tag{65}$$

G-Mean equally accounts the recall value of each class, so it measures the balanced performance of classifier. Therefore, in this research work, we have compared the performance of proposed WMLSTSVM classifier with other approaches by using G-Mean.

7.3. *Statistical tests for performance comparison*

In this research work, hypothesis testing techniques such as Wilcoxon signed rank and Friedman statistic tests are used to provide statistical support for the analysis of the results. Wilcoxon signed rank test is a non-parametric statistical technique that perform pairwise comparisons between two classifiers [14, 19, 82]. In this technique, the difference between the performances of two classifiers is computed for each dataset. It ranks the absolute differences from smallest to largest and average ranks are assigned in case of ties. The rank R^+ stores the sum of ranks for the datasets on which our proposed classifier outperformed the other classifiers, and rank R^- stores the sum of ranks for the opposite case. Wilcoxon signed rank test follows z-distribution. Consider 'T' to be the smaller of the R^+ and R^- , $T=\min(R^+, R^-)$. If T is less than or equal to the Wilcoxon distribution, the null hypothesis which states that there is no difference between the classifiers can be rejected. It is also very useful to compute the p-value associated with each comparison as it represents the lowest level of significance of a hypothesis that results in rejection. By doing this, we can find whether two classifiers are significantly different and the manner in which they are different.

On the other hand, Friedman test ranks the algorithms according to their performance for each dataset separately, the best performing algorithm gets the rank of 1, the second best rank 2 and so on [19,83]. Average ranks are assigned in case of ties. Let r_i^j be the rank of the j^{th} of M classifiers on the i^{th} of N datasets. Friedman test statistic is computed as:

$$\chi_F^2 = \frac{12N}{M(M+1)}\left[\sum_{j=1}^{M} AR_j^2 - \frac{M(M+1)^2}{4}\right], \tag{66}$$

where $AR_j = \frac{1}{N}\sum_{i=1}^{N} r_i^j$. Friedman test statistic is distributed according to the Chi-square distribution with M-1 degrees of freedom. If the value of χ_F^2 is large enough, then the null hypothesis can be rejected. The significant differences between individual classifiers are tested by using post hoc Nemenyi test [84]. According to this, if the average rank of two algorithms differs by at least the critical difference, then these algorithms are significantly different. Critical difference (CD) is defined as:

$$CD = q_\alpha \sqrt{\frac{M(M+1)}{6N}} \tag{67}$$

where $\boldsymbol{q_\alpha}$ is based on the Studentized range statistic. The results from Friedman and Nemenyi post hoc tests are plotted and visualized by using a modified version of Demsar significance diagram [85].

7.4. *Experiment and result Analysis*

We have evaluated the performance of the proposed WMLSTSVM classifier with other classifiers such as Multi-SVM, AdaBoost.NC, Multiple Birth Twin Support Vector Machine (MBSVM) [81], OVO-MLSTSVM and OVA MLSTSVM using 10 fold cross validation. Multi-SVM and MBSVM classifiers are combined with pre-processing approaches such as

Static-SMOTE, Global-CS and Random Oversampling (ROS). Each classifier is implemented in matlabR2012a on a Windows based operating system with Intel Core i-7 processor (3.4 GHz) with 12-GB RAM. Grid search approach is used for parameters selection. Penalty parameters are selected from the set $\{10^{-8},...,10^{5}\}$ while sigma parameter for Gaussian kernel function is chosen from the set $\{2^{-5},...,2^{10}\}$. Since, G-Mean measures the balanced performance of a classifier, therefore, for imbalanced dataset, G-Mean is a good choice for evaluation. Table 3 and 4 present the performance comparison of the proposed classifier with existing classifiers including the average of G-Means (GM), standard G-Means and time (including both training and testing) of 10-folds. Bold values indicate better performance of the classifier. For linear cases, G-Mean of the proposed classifier is better for each dataset except Hayes Roth, Lymphography, Zoo and Splice. Adaboost.NC has shown better performance for Lympography and Zoo dataset. For non-linear cases, WMLSTSVM has obtained better G-Mean than that of other nine classifiers for 12 out of 15 datasets. It is also observed that the proposed classifier takes comparable computation time on almost all type of datasets as compared to other classifiers. Therefore, it could be concluded that the proposed WMLSTSVM classifier has the highest computation efficiency.

Table 3. Geometric Mean Comparison of linear classifiers on benchmark datasets

Data set	Static SMOTE SVM GM±std(%) Time(s)	Global-CS SVM GM±std(%) Time(s)	ROS SVM GM±std(%) Time(s)	Adaboost.NC GM±std(%) Time(s)	Static-SMOTE MBSVM GM±std(%) Time(s)	Global-CS MBSVM GM±std(%) Time(s)	ROS MBSVM GM±std(%) Time(s)	OVO MLSTSVM GM±std(%) Time(s)	OVA MLSTSVM GM±std(%) Time(s)	WMLSTSVM GM±std(%) Time(s)
Bal	79.64±5.29 4.55	78.38±6.15 5.36	82.60±5.45 5.22	85.81±5.02 4.38	84.27±5.15 0.428	83.82±5.26 0.254	81.93±4.76 0.491	76.11±3.97 0.089	85.51±4.35 0.07176	**89.59±3.86** 0.127
Eco	67.95±5.43 2.38	70.89±5.47 3.24	73.04±5.72 3.35	74.58±4.64 3.16	76.75±5.2 1.08	72.63±5.37 1.127	71.45±5.21 1.18	66.40±4.27 0.0803	74.02±5.13 0.07086	**80.86±5.02** 0.1029
Gls	62.19±4.14 3.04	59.09±6.65 2.35	64.47±5.88 2.39	55.62±6.17 2.65	54.61±5.62 0.088	66.67±6.08 0.052	62.19±5.32 0.0949	**74.72±3.37** 0.00481	61.05±4.58 0.00468	64.33±4.24 0.0926
Win	91.08±3.28 0.411	91.63±5.73 0.097	92.68±3.65 0.57	94.30±2.54 0.24	96.49±3.08 0.0102	95.38±2.96 0.014	95.07±1.95 0.0382	100±0.0 0.0085	100±0.0 0.0078	**100±0.0** 0.00858
Thy	91.24±3.13 0.664	89.14±5.02 0.478	88.32±3.83 0.568	91.67±3.25 0.85	92.04±3.72 0.018	94.73±3.04 0.0093	91.79±2.36 0.0803	100±0.0 0.0056	100±0.0 0.00624	**100±0.0** 0.0135
HaR	64.83±4.27 2.87	65.47±5.23 2.126	60.95±4.6 2.18	57.82±4.16 1.96	**71.08±4.53** 0.01283	67.45±4.22 0.0096	66.85±4.28 0.0218	52.38±4.04 0.00742	69.19±4.19 0.00468	70.94±3.58 0.005304
Der	82.74±4.15 2.96	85.62±4.83 3.45	82.74±5.02 3.664	91.13±4.58 3.48	83.43±4.67 0.108	82.14±4.5 0.09775	87.32±3.73 0.0928	87.22±4.65 0.0923	88.70±4.84 0.0702	**92.89±3.87** 0.0907
Shu	65.27±3.65 7.24	73.48±5.75 6.8	67.94±4.53 7.46	80.32±5.02 6.33	78.02±4.84 0.9577	82.43±4.83 1.118	76.11±3.97 1.26	80.69±4.22 0.2169	84.56±4.91 0.14196	**92.26±3.12** 0.4577
PnB	74.58±4.60 5.82	78.52±6.88 5.509	72.32±4.46 5.26	78.53±4.84 4.46	81.82±4.22 0.1293	85.51±5.03 0.1928	78.90±4.32 0.218	85.87±3.23 0.283	82.19±2.56 0.1045	**88.71±2.87** 0.347
PgB	83.87±5.59 5.35	70.28±5.27 5.67	74.08±5.24 5.45	81.52±5.27 5.02	77.31±5.24 1.562	85.63±4.26 1.1499	79.85±4.29 1.25	83.39±4.42 0.0874	85.77±4.12 0.08112	**86.29±4.34** 0.09048
Con	34.82±4.78 5.02	48.83±5.39 4.56	40.92±5.29 4.38	49.01±4.13 4.86	43.08±5.66 0.5624	42.78±5.48 0.4022	45.67±4.41 0.6091	48.03±4.33 0.1302	45.18±4.46 0.10764	**54.62±4.95** 0.13752
Lym	80.25±4.46 2.41	77.49±4.72 2.11	78.74±4.78 2.63	**83.35±5.62** 2.24	82.04±4.42 0.486	79.87±4.10 0.6246	79.51±4.65 0.4302	78.46±3.87 0.0959	80.54±4.28 0.0904	81.69±4.16 0.112
Zoo	91.78±3.65 0.03	89.84±4.26 0.0265	91.88±3.97 0.0382	**93.82±3.24** 0.048	92.67±3.25 0.012	92.51±4.67 0.035	91.46±3.83 0.0545	89.85±3.73 0.0068	89.32±4.05 0.00624	93.5±3.98 0.007452
Spl	84.25±3.67 10.35	77.88±3.57 10.14	81.52±3.88 11.04	90.05±3.02 12.53	**91.08±3.34** 4.45	84.39±3.63 4.78	85.87±3.22 4.28	78.12±4.38 3.52	84.72±4.73 3.78	90.67±3.88 4.02
Cld	30.74±4.80 2.33	30.22±4.45 2.02	28.08±3.67 2.11	33.82±3.16 3.04	33.62±3.16 1.16	35.86±4.56 0.9246	34.52±3.33 0.988	29.88±3.69 1.03	32.06±4.17 0.9048	**39.85±4.34** 1.08

Table 4. Geometric Mean Comparison of non-linear classifiers on benchmark datasets

Data set	Static SMOTE SVM GM±std(%) Time(s)	Global-CS SVM GM±std(%) Time(s)	ROS SVM GM±std(%) Time(s)	Adaboost.NC GM±std(%) Time(s)	Static-SMOTE MBSVM GM±std(%) Time(s)	Global-CS MBSVM GM±std(%) Time(s)	ROS MBSVM GM±std(%) Time(s)	OVO MLSTSVM GM±std(%) Time(s)	OVA MLSTSVM GM±std(%) Time(s)	WMLSTSVM GM±std(%) Time(s)
Bal	86.50±6.39 8.03	87.45±6.03 7.65	85.32±6.46 8.57	87.24±5.17 8.65	88.64±5.53 3.28	88.53±5.68 2.85	89.18±5.66 3.16	74.29±5.36 0.872	90.64±4.84 0.9188	**92.83±3.9** 0.9356
Eco	76.29±6.17 3.56	75.61±5.91 3.82	77.87±5.92 4.22	80.18±5.02 4.24	80.06±5.45 1.21	79.77±6.24 1.16	81.03±5.82 1.48	69.62±5.8 0.663	81.34±5.78 0.3962	**87.06±4.77** 0.4003
Gls	59.32±6.26 7.11	63.76±6.08 8.01	62.17±6.06 8.48	63.64±6.69 7.83	65.29±5.88 2.49	70.36±6.13 2.52	69.84±6.26 3.02	83.26±5.03 0.5116	76.63±5.35 0.2249	**88.36±3.62** 0.2256
Win	93.61±3.15 0.9059	96.14±2.89 0.8892	95.35±3.01 1.05	96.03±2.53 1.27	97.22±2.7 0.3386	98.02±2.65 0.375	97.77±2.66 0.367	100±0.0 0.08892	100±0.0 0.092	**100±0.0** 0.106
Thy	95.63±2.58 2.34	95.46±3.04 2.11	90.91±3.74 2.45	95.22±2.86 3.04	94.73±3.49 1.02	97.84±2.88 0.9204	97.18±3.17 1.12	100±0.0 0.120	100±0.0 0.132	**100±0.0** 0.155
HaR	62.74±4.86 6.88	64.21±4.63 7.24	63.70±4.58 6.92	64.89±5.04 7.16	69.91±4.22 1.97	70.24±3.56 2.083	65.83±4.51 2.06	54.55±4.37 2.25	68.07±4.95 1.369	**85.49±3.53** 1.568
Der	85.42±5.81 8.78	89.28±5.43 8.63	86.23±5.75 9.1	90.08±5.69 8.59	91.45±4.34 3.72	94.82±3.92 4.02	90.01±6.03 3.89	91.67±4.23 1.135	92.77±4.2 0.5538	**97.00±4.14** 0.736
Shu	67.29±4.66 20.86	76.85±4.39 18.98	65.76±5.02 24.57	77.87±4.67 20.24	84.26±4.12 6.88	80.16±5.43 5.74	75.42±5.32 5.85	82.74±4.85 1.56	83.11±4.11 2.24	**94.97±2.96** 3.12
PnB	78.25±5.22 30.27	82.15±5.16 32.62	75.73±5.89 30.15	83.98±5.37 31.67	80.25±5.56 12.56	83.27±4.35 12.84	81.28±4.78 13.23	87.16±3.66 11.01	89.91±2.93 10.86	**95.19±2.35** 10.94
PgB	85.34±6.28 11.77	76.82±6.34 11.21	77.67±6.67 12.38	86.29±6.02 14.02	87.14±5.05 4.764	85.98±5.66 4.2	85.02±6.12 4.305	81.67±6.23 1.719	87.12±4.88 1.677	**89.87±4.62** 1.89
Con	40.66±5.39 29.01	46.98±4.82 28.37	45.26±5.11 28.08	46.18±4.86 26.08	49.01±4.82 10.42	**56.16±5.01** 9.48	53.18±5.37 10.24	46.24±4.7 5.53	48.33±4.5 5.6	55.38±4.85 6.03
Lym	82.33±4.53 5.47	81.04±4.89 6.59	83.02±4.69 6.85	82.74±5.24 5.57	83.52±4.35 1.14	81.21±5.13 1.62	82.60±5.6 1.5	80.28±4.85 0.2534	82.96±4.93 0.2277	**85.12±3.55** 0.2418
Zoo	96.04±3.02 0.932	94.45±3.67 0.9859	93.22±4.01 1.36	95.35±3.53 1.10	**96.92±3.65** 0.384	95.18±4.4 0.5621	94.07±4.02 0.674	93.67±4.86 0.08112	94.32±3.74 0.0665	94.37±3.74 0.08736
Spl	89.88±3.77 42.45	81.41±3.48 44.22	80.34±3.62 42.11	**95.73±3.34** 36.54	94.05±3.03 16.26	88.28±3.68 16.29	87.67±4.25 15.32	80.55±5.04 8.02	91.02±3.92 7.66	93.85±3.38 7.85
Cld	34.56±4.82 7.4	32.34±4.79 6.36	30.57±5.11 6.94	35.56±5.11 5.59	35.63±4.66 2.75	34.09±5.02 2.56	32.72±4.61 3.11	32.69±4.7 1.24	34.60±4.96 1.38	**39.08±4.26** 1.56

Wilcoxon test is used for the comparison between linear and non-linear classifiers. In table 5, we have compared the performance of each classifier for both linear and non-linear cases. The rank for non-linear cases is indicated by R^+ and for linear cases by R^-. This test concludes that the non-linear classifiers are statistically better than the linear classifiers in all the cases of study with a high degree of confidence. It is observed from the table that the p-value is less than 0.05 in all the cases.

Table 5. Wilcoxon test for the comparison between linear and non-linear classifiers

Classifiers	R^+(non-linear)	R^-(linear)	p-value
Static Smote SVM	109.0	11.0	0.0056
Global-CS SVM	117.0	3.0	0.0013
ROS SVM	112.0	8.0	0.0033
AdaBoost.NC	102.0	18.0	0.0178
Static Smote MBSVM	114.0	6.0	0.0023
Global-CS MBSVM	110.0	10.0	0.0047
ROS MBSVM	113.5	6.5	0.0025
OVO MLSTSVM	81.5	9.5	0.0124
OVA MLSTSVM	84.0	7.0	0.0076
WMLSTSVM	89.0	2.0	0.0025

The results obtained from the two statistical tests are shown in table 6. For Wilcoxon test, we have compared the performance of the proposed classifier with other classifiers and calculated the ranks and p-value for each case. From the results, it is concluded that the proposed WMLSTSVM classifier outperforms all of them with high degree of confidence. Friedman test is also applied and the average rank of each classifier is calculated according to their G-Mean as shown in table 6. It can be concluded that the WMLSTSVM has the highest average rank among all classifiers. Friedman test statistic is calculated for both linear and non-linear cases according to equation 66. In both the cases its value is very high from the critical value for 9 –degree of freedom which is 16.9190. Therefore, the null hypothesis which states that there is no difference between the classifiers is rejected.

Table 6. Result of Wilcoxon signed rank test and Friedman test

Wilcoxon signed rank test						
	Linear			Non-linear		
Classifiers	R^+	R^-	p-value	R^+	R^-	p-value
WMLSTSVM-OVA MLSTSVM	91	0	0.0016	91	0	0.0016
WMLSTSVM-OVO MLSTSVM	83	8	0.0093	91	0	0.0016
WMLSTSVM-ROS MBSVM	120	0	0.0007	120	0	0.0007
WMLSTSVM-Global CS MBSVM	116	4	0.0015	117	3	0.0013
WMLSTSVM-Static SMOTE MBSVM	114	6	0.0023	116	4	0.0015
WMLSTSVM-AdaBoost.NC	116	4	0.0015	117	3	0.0013
WMLSTSVM-ROS SVM	119	1	0.0008	120	0	0.0007
WMLSTSVM- Global CS SVM	120	0	0.0007	119	1	0.0008
WMLSTSVM-Static SMOTE SVM	120	0	0.0007	119	1	0.0008

Friedman Test						
		Linear			Non-linear	
Classifiers	Mean Rank	p-value	χ_F^2	Mean Rank	p-value	χ_F^2
WMLSTSVM	1.67			1.67		
OVA MLSTSVM	4.33			3.33		
OVO MLSTSVM	6.00			6.53		
ROS MBSVM	5.76			5.73		
Global CS MBSVM	4.73	<0.05	55.37	4.46	<0.05	69.29
Static SMOTE MBSVM	4.67			4.06		
AdaBoost.NC	4.53			5.4		
ROS SVM	7.76			8.8		
Global CS SVM	8.0			7.4		
Static SMOTE SVM	7.53			7.6		

Critical difference (CD) for α=0.05 is:

$$CD=3.164\sqrt{\frac{10\times11}{6\times15}}=3.49$$

Critical value $q_{0.05}$ for 10 classifiers is 3.164[19]. Figure 2 shows the significance diagram in which classifiers are listed in ascending order of ranked performance on the y-axis and the classifier's AR across all fifteen datasets on the x-axis. Two vertical lines represent the difference of end of the best performing approach's tail and the start of the next significantly different approach.

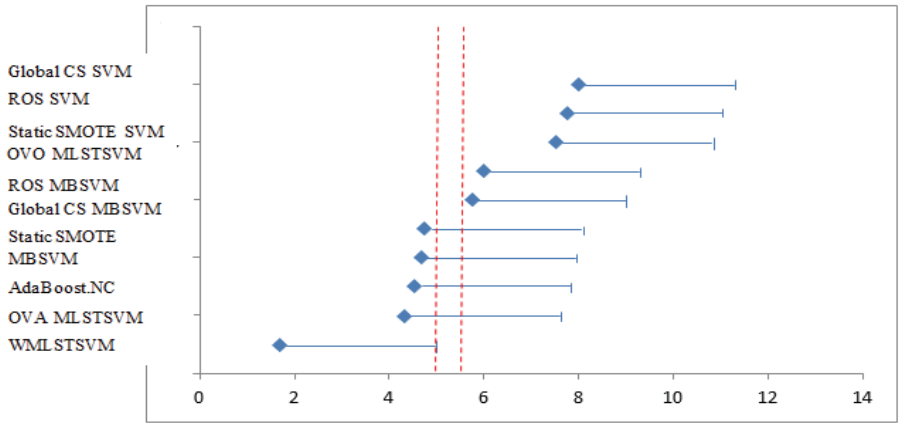

Fig. 2. Average rank comparison of linear classifiers

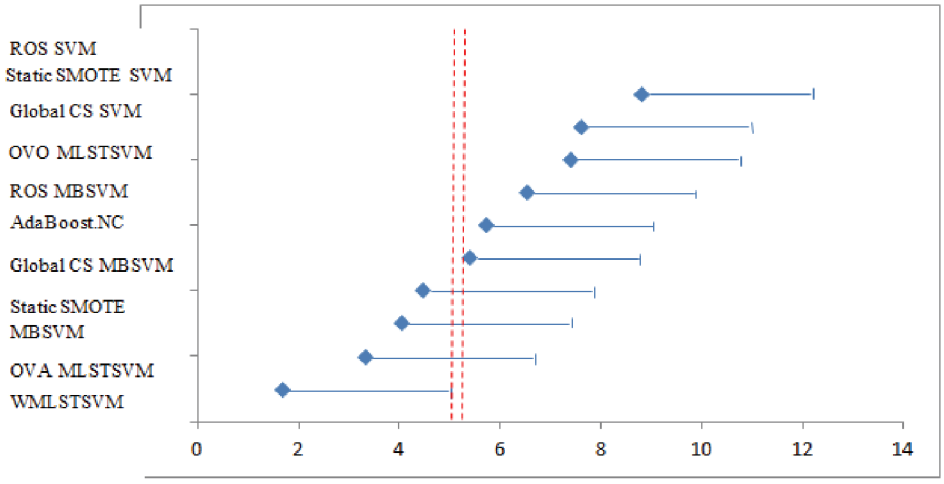

Fig. 3. Average rank comparison of non-linear classifiers

The diagram clearly depicts that there is no significant difference between WMLSTSVM, OVA-MLSTSVM, Global-CS MBSVM, Static SMOTE MBSVM and Adaboost.NC for linear cases. However, ROS MBSVM, OVO MLSTSVM, Static SMOTE SVM, ROS SVM and Global-CS SVM classifiers perform significantly worse than the best performing classifier with values 5.76, 6, 7.53, 7.76 and 8. Figure 3 shows the average rank comparison of classifiers for non-linear cases. It is observed that WMLSTSVM performs significantly better than Adaboost.NC, ROS MBSVM, OVO MLSTSVM, Global CS SVM, Static SMOTE SVM and ROS SVM classifiers for non-linear cases. So, we can conclude that the proposed WMLSTSVM classifier performs significantly better in case of multi-class imbalanced problem scenario.

8. Conclusion

Most of the solutions to class imbalance problem concentrate on two-class setting. The two-class imbalance solutions are not directly applicable to multi-class scenario. This research work addressed the imbalance problem in multi-class scenario and proposed a novel classifier named as WMLSTSVM. Firstly, a novel multi-classifier MLSTSVM is proposed which is the multi-class extension of the binary LSTSVM. Then, appropriate weight setting is done in loss function to control the sensitivity of the classifier for imbalanced data in determining each hyper-plane. The validity of the proposed approach has been proved on fifteen benchmark datasets which are imbalanced in nature. Statistical analysis of the performance of each classifier also confirms that the WMLSTSVM classifier is the best performing classifier and is a suitable choice for handling imbalanced data problem in multi-class scenario. For future work, it would be interesting to select the parameters by using Genetic Algorithm or Particle Swarm Optimization and investigate the performance of WMLSTSVM with real world data.

References

1. N. Japkowicz and S. Stephen. "The class imbalance problem: A systematic study." *Intelligent data analysis* 6, no. 5 (2002), pp. 429-449.
2. N.V. Chawla, N. Japkowicz, and A. Kotcz. "Editorial: special issue on learning from imbalanced data sets." *ACM Sigkdd Explorations Newsletter* 6, no. 1 (2004), pp. 1-6.
3. Y. Sun, A. K. Wong, and M. S. Kamel. "Classification of imbalanced data: A review." *International Journal of Pattern Recognition and Artificial Intelligence* 23, no. 04 (2009), pp. 687-719.
4. M. Kubat, R. C. Holte, and S. Matwin. "Machine learning for the detection of oil spills in satellite radar images." *Machine learning* 30, no. 2-3 (1998), pp. 195-215.
5. Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang. "Cost-sensitive boosting for classification of imbalanced data." *Pattern Recognition* 40, no. 12 (2007), pp. 3358-3378.
6. Y. Sun, A. K. Wong, and M. S. Kamel. "Classification of imbalanced data: A review." International Journal of Pattern Recognition and Artificial Intelligence 23, no. 04 (2009), pp. 687-719.
7. T. Fawcett and F. Provost. "Adaptive fraud detection." *Data mining and knowledge discovery* 1, no. 3 (1997), pp. 291-316.
8. K. Bache and M. Lichman, "UCI Machine Learning Repository" [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, (2013).

9. K.J. Ezawa, M. Singh, and S. W. Norton. "Learning goal oriented Bayesian networks for telecommunications risk management." In *ICML*, (1996), pp. 139-147.
10. C. Cardie and N. Howe. "Improving minority class prediction using case-specific feature weights." In *ICML*, (1997), pp. 57-65.
11. A. Orriols-Puig and E. Bernadó-Mansilla. "Evolutionary rule-based systems for imbalanced data sets." Soft Computing 13, no. 3 (2009), pp. 213-225.
12. S. Kotsiantis, D. Kanellopoulos and P. Pintelas. "Handling imbalanced datasets: A review." *GESTS International Transactions on Computer Science and Engineering* 30, no. 1 (2006), pp. 25-36.
13. S. Wang and X. Yao. "Multiclass imbalance problems: Analysis and potential solutions." Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 42, no. 4 (2012), pp. 1119-1130.
14. A. Fernández, V. López, M. Galar, M. A.J. Del Jesus, and F. Herrera. "Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches." Knowledge-Based Systems 42 (2013), pp. 97-110.
15. G.E. Batista, R. C. Prati, and M.C. Monard. "A study of the behavior of several methods for balancing machine learning training data." *ACM Sigkdd Explorations Newsletter* 6, no. 1 (2004), pp. 20-29.
16. A. Estabrooks, T. Jo and N. Japkowicz. "A multiple resampling method for learning from imbalanced data sets." *Computational Intelligence* 20, no. 1 (2004), pp. 18-36.
17. He, Haibo, and Edwardo A. Garcia. "Learning from imbalanced data." *Knowledge and Data Engineering, IEEE Transactions on* 21, no. 9 (2009), pp. 1263-1284.
18. B. Zadrozny and C. Elkan. "Learning and making decisions when costs and probabilities are both unknown." *In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, (2001), pp. 204-213.
19. J. Demšar. "Statistical comparisons of classifiers over multiple data sets." *The Journal of Machine Learning Research* 7 (2006), pp. 1-30.
20. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. "SMOTE: synthetic minority over-sampling technique." *arXiv preprint arXiv:1106.1813* (2011).
21. A. Fernández, S. García, M. JD Jesus and F. Herrera. "A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets." *Fuzzy Sets and Systems 159*, no. 18 (2008), pp. 2378-2398.
22. J. Derrac, I. Triguero, C. J. Carmona and F. Herrera. "Evolutionary-based selection of generalized instances for imbalanced classification." *Knowledge-Based Systems* 25, no. 1 (2012): 3-12.
23. M.A. Tahir, J. Kittler and F. Yan. "Inverse random under sampling for class imbalance problem and its application to multi-label classification." *Pattern Recognition* 45, no. 10 (2012), pp. 3738-3750.
24. Y. Tang, Y. Q. Zhang, N. V. Chawla, and S. Krasser. "SVMs modeling for highly imbalanced classification." *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 39, no. 1 (2009), pp. 281-288.
25. A. Estabrooks, T. Jo and N. Japkowicz. "A multiple resampling method for learning from imbalanced data sets." *Computational Intelligence* 20, no. 1 (2004), pp. 18-36.
26. J. Laurikkala, "Instance-based data reduction for improved identification of difficult small classes." *Intelligent Data Analysis* 6, no. 4 (2002), pp. 311-322.
27. J. M. Choi. "A selective sampling method for imbalanced data learning on support vector machines." (2010).
28. Y. Liu, X. Yu, J. X. Huang, and A. An. "Combining integrated sampling with SVM ensembles for learning from imbalanced datasets." *Information Processing & Management* 47, no. 4 (2011), pp. 617-631.
29. X. Y. Liu, J. Wu, and Z. H. Zhou. "Exploratory undersampling for class-imbalance learning." *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 39, no. 2 (2009), pp. 539-550.
30. C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap. "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem." *In Advances in Knowledge Discovery and Data Mining*, Springer Berlin Heidelberg (2009), pp. 475-482.
31. I. Tomek. "Two modifications of CNN." *IEEE Trans. Systems, Man and Cybernetics 6* (1976), pp. 769-772.
32. M. Kubat and S. Matwin. "Addressing the curse of imbalanced training sets: one-sided selection." *In ICML,* vol. 97, (1997), pp. 179-186.
33. T. Jo and N. Japkowicz. "Class imbalances versus small disjuncts." *ACM SIGKDD Explorations Newsletter* 6, no. 1 (2004), pp. 40-49.
34. S. J. Yen and Y.S. Lee. "Cluster-based under-sampling approaches for imbalanced data distributions." *Expert Systems with Applications* 36, no. 3 (2009), pp. 5718-5727.
35. S. Chen, G. Guo and L. Chen. "A new over-sampling method based on cluster ensembles." *In Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*, (2010), pp. 599-604.
36. R. Barandela, J. S. Sánchez, V. Garcıa and E. Rangel. "Strategies for learning in class imbalance problems." *Pattern Recognition* 36, no. 3 (2003), pp. 849-851.
37. C. Diamantini and D. Potena. "Bayes vector quantizer for class-imbalance problem." *Knowledge and Data*

Engineering, IEEE Transactions on 21, no. 5 (2009), pp. 638-651.

38. D. A. Cieslak, T. R. Hoens, N. V. Chawla, and W. P. Kegelmeyer. "Hellinger distance decision trees are robust and skew-insensitive." *Data Mining and Knowledge Discovery* 24, no. 1 (2012), pp. 136-158.
39. N. G.Pedrajas, J P. Rodríguez, M. G. Pedrajas, D. O. Boyer, and C. Fyfe. "Class imbalance methods for translation initiation site recognition in DNA sequences." *Knowledge-Based Systems* 25, no. 1 (2012), pp. 22-34.
40. B. Zadrozny and C. Elkan. "Learning and making decisions when costs and probabilities are both unknown." *In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, (2001), pp. 204-213.
41. L. Mena and J. A. Gonzalez. "Symbolic one-class learning from imbalanced datasets: application in medical diagnosis." *International Journal on Artificial Intelligence Tools* 18, no. 02 (2009), pp. 273-309.
42. D. M. Tax, "One-class classification". PhD thesis, Delft University of Technology, 2001.
43. B. X. Wang and N. Japkowicz. "Boosting support vector machines for imbalanced data sets." *Knowledge and Information Systems* 25, no. 1 (2010), pp. 1-20.
44. C. Chen, A. Liaw and L. Breiman. "Using random forest to learn imbalanced data." *University of California, Berkeley* (2004).
45. C. Elkans. "The foundations of cost-sensitive learning." *In International joint conference on artificial intelligence*, vol. 17, no. 1, pp. 973-978. LAWRENCE ERLBAUM ASSOCIATES LTD, 2001.
46. K. M. Ting. "An instance-weighting method to induce cost-sensitive trees." *Knowledge and Data Engineering, IEEE Transactions on* 14, no. 3 (2002), pp. 659-665.
47. B. Zadrozny, J. Langford and N. Abe. "Cost-sensitive learning by cost-proportionate example weighting." In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pp. 435-442. IEEE, 2003.
48. J. M. örg. "Classifier Learning for Imbalanced Data with Varying Misclassification Costs." (2006).
49. R. Akbani, S. Kwek and N. Japkowicz. "Applying support vector machines to imbalanced datasets." In *Machine Learning: ECML 2004*, pp. 39-50. Springer Berlin Heidelberg, 2004.
50. X. Yang, Q. Song, and Y. Wang. "A weighted support vector machine for data classification." *International Journal of Pattern Recognition and Artificial Intelligence* 21, no. 05 (2007), pp. 961-976.
51. D. Tomar, S. Singhal, and S. Agarwal. "Weighted Least Square Twin Support Vector Machine for Imbalanced Dataset." *International Journal of Database Theory & Application* 7, no. 2 (2014).
52. Z. H. Zhou and X. Y. Liu. "Training cost-sensitive neural networks with methods addressing the class imbalance problem." *Knowledge and Data Engineering, IEEE Transactions on* 18, no. 1 (2006), pp. 63-77.
53. X.Y. Liu and Z. H. Zhou. "The influence of class imbalance on cost-sensitive learning: An empirical study." In Data Mining, 2006. ICDM'06. Sixth International Conference on, pp. 970-974. IEEE, 2006.
54. K. McCarthy, B. Zabar, and G. Weiss. "Does cost-sensitive learning beat sampling for classifying rare classes?." In Proceedings of the 1st international workshop on Utility-based data mining, pp. 69-77. ACM, 2005.
55. M.A. Maloof. "Learning when data sets are imbalanced and when costs are unequal and unknown." *In ICML-2003 workshop on learning from imbalanced data sets II*, vol. 2, (2003), pp. 2-1.
56. L. Breiman, J. Friedman, C. J. Stone and R. A. Olshen. *Classification and regression trees. CRC press*, 1984.
57. C. Drummond and R. C. Holte. "Exploiting the cost (in) sensitivity of decision tree splitting criteria." *In ICML,* (2000), pp. 239-246.
58. M. Kukar and I. Kononenko. "Cost-Sensitive Learning with Neural Networks." *In ECAI*, (1998), pp. 445-449.
59. W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. "AdaCost: misclassification cost-sensitive boosting." *In ICML,* (1999), pp. 97-105.
60. P. Domingos and M. Pazzani. "Beyond independence: Conditions for the optimality of the simple bayesian classi er." *In Proc. 13th Intl. Conf. Machine Learning*, (1996), pp. 105-112.
61. G. I. Webb and M. J. Pazzani. "Adjusted probability naive Bayesian induction." *In Advanced Topics in Artificial Intelligence*, Springer Berlin Heidelberg, (1998), pp. 285-295.
62. R. Kohavi and D. H. Wolpert. "Bias plus variance decomposition for zero-one loss functions." *In ICML*, (1996), pp. 275-283.
63. J. Gama. "Iterative bayes." *Theoretical Computer Science* 292, no. 2 (2003), pp. 417-430.
64. G. Fumera and F. Roli. "Support vector machines with embedded reject option." In Pattern Recognition with Support Vector Machines, Springer Berlin Heidelberg, (2002), pp. 68-82.
65. J.C. Platt. "Fast training of support vector machines using sequential minimal optimization." In Advances in kernel methods, MIT Press, (1999), pp. 185-208.
66. J.Y. Kwok. "Moderating the outputs of support vector machine classifiers." *Neural Networks, IEEE Transactions on* 10, no. 5 (1999), pp. 1018-1031.
67. F.F.Navarro, C. H. Martínez, and P. A. Gutiérrez. "A dynamic over-sampling procedure based on sensitivity for multi-class problems." *Pattern Recognition* 44, no. 8 (2011), pp.1821-1833.

68. Z. H. Zhou and X.Y. Liu. "ON MULTI-CLASS COST-SENSITIVE LEARNING." *Computational Intelligence* 26, no. 3 (2010), pp. 232-257.
69. A. Tan, D. Gilbert and Y. Deville. "Multi-class protein-fold classification using a new ensemble machine learning approach." (2003).
70. X.M. Zhao, X. Li, L. Chen and K. Aihara. "Protein classification with imbalanced data." *Proteins: Structure, function, and bioinformatics* 70, no. 4 (2008),pp. 1125-1132.
71. K. Chen, B. L. Lu and J. T. Kwok. "Efficient classification of multi-label and imbalanced data using min-max modular classifiers." *In Neural Networks*, 2006. IJCNN'06. International Joint Conference on, IEEE, (2006), pp. 1770-1775.
72. T. W. Liao. "Classification of weld flaws with imbalanced class data." Expert Systems with Applications 35, no. 3 (2008), pp. 1041-1052.
73. A. Fernández, M. JD Jesus, and F. Herrera. "Multi-class imbalanced data-sets with linguistic fuzzy rule based classification systems based on pairwise learning." *In Computational Intelligence for Knowledge-Based Systems Design*, Springer Berlin Heidelberg, (2010), pp. 89-98.
74. P. Domingos. "Metacost: A general method for making classifiers cost-sensitive." In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, (1999), pp. 155-164.
75. Y. Sun, M. S. Kamel and Y. Wang. "Boosting for Learning Multiple Classes with Imbalanced Class Distribution." In *ICDM*, vol. 6, (2006), pp. 592-602.
76. Jayadeva, R. Khemchandani, S. Chandra, Twin Support Vector Machine for pattern classification. IEEE Trans Pattern Anal Mach Intell, 29(5), (2007), pp.905–910.
77. C. Cortes and V. Vapnik. "Support-vector networks." Machine learning 20, no. 3, (1995), pp.273-297.
78. OL Mangasarian, EW Wild, "Multisurface proximal support vector classification via generalized eigenvalues". *IEEE Trans Pattern Anal Mach Intell,* 28(1), 2006, pp.69–74.
79. M. A. Kumar and M. Gopal, "Least squares twin support vector machines for pattern classification", Expert Systems with Applications 36, (2009), pp. 7535–7543.
80. J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez and F. Herrera. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing 17:2-3, (2011), 255-287.
81. Z. X. Yang, Y.H. Shao and X.S.Zhang. "Multiple birth support vector machine for multi-class classification." Neural Computing and Applications 22, no. 1, 2013, pp. 153-161.
82. D.Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, second ed., Chapman & Hall, CRC 2006.
83. M. Friedman. "A comparison of alternative tests of significance for the problem of m rankings." *The Annals of Mathematical Statistics* 11, no. 1 (1940), pp. 86-92.
84. P. Nemenyi. Distribution-free multiple comparisons. Ph.D. Thesis. Princeton University, 1963.
85. S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. "Benchmarking classification models for software defect prediction: A proposed framework and novel findings." Software Engineering, IEEE Transactions on 34, no. 4 (2008), pp. 485-496.