Machine Learning for Mining Imbalanced Data

Md. Yasir Arafat, Sabera Hoque, Shuxiang Xu and Dewan Md Farid

***Abstract*—Mining imbalanced data, which is also known as a class imbalanced problem is one of the most enormously challenging tasks in machine learning for data mining applications. To achieve overall accurate performance in imbalanced classification employing machine learning techniques is difficult as the majority class instances always overpower the minority class instances by a huge difference. An unequal distribution is very common in real-world high dimensional datasets, where binary classification is more frequent than multi-class classification task. Most existing machine learning algorithms are more focused on classifying majority class instances while ignoring or misclassifying minority class instances. Several techniques have been introduced in the last decades for imbalanced data classification, where each of this techniques has their own advantages and disadvantages. In this paper, we have studied and compared 12 extensively imbalanced data classification methods: SMOTE, AdaBoost, RUSBoost, EUSBoost, SMOTEBoost, MSMOTEBoost, DataBoost, Easy Ensemble, BalanceCascade, OverBagging, UnderBagging, SMOTEBagging to extract their characteristics and performance on 27 imbalanced datasets. In general, the combination of both over-sampling and under-sampling techniques with ensemble classifiers such as bagging and boosting achieve the highest accuracy for classifying both majority and minority class instances. Additionally, an extensive and critical review of the existing algorithms of imbalanced learning is provided with detailed discussion. According to our findings, we advise some practical suggestions based on the reviewed papers to offer further research directions for imbalanced learning.**

***Index Terms*—Classification; Data Sampling; Ensemble Learning; Imbalanced Data; Machine Learning; Random Sampleing.**

I. Introduction

Imbalanced data classification is an important problem in supervised learning, where the classes are not represented fairly, i.e., one class is underwhelmed called as minority class while the other class is overwhelmed called as majority class. In many real-world applications, imbalanced datasets are very commonly seen in medical diagnosis, fraud detection, anomaly detection, managing risk and predicting failures of technical equipment, defected product on manufacturing, etc. For example, consider a dataset that consists of 100,000 patient records of a hospital, where a list of candidates who tested for diagnosis of cancer. It was found in the dataset that 98% patients did not have cancer and only 2% got unlucky. As another example, in credit card fraud detection, fraudulent transactions will occur very rare in comparison with regular transactions. Imbalanced datasets are often considered imbalanced to mean a minority class of 10% to 20%. In reality, datasets can get far more imbalanced than this, for example, about 2% of credit card accounts are defrauded every year; Medical screening for a situation is generally performed on an enormous community without the condition, to detect a finite minority with it (e.g., HIV prevalence in the USA is 0.4%). To mine the real-world datasets, classifiers are often biased with significant class imbalance problem. The conventional machine learning and data mining algorithms classify the majority class instances more accurately but fail to classify the minority class instances [1]. As the high-dimensional real-world data is used mostly by conventional machine learning algorithms so it is often become a difficult task to classify them properly. Consequently, the over-generalization on the majority class leads a poor performance on the minority class. To resolve this problem of mining imbalanced data various methods have been proposed in the last decade by various researchers: such as sampling methods, cost-sensitive methods, and ensemble learning methods [2]. Sampling can be done in two different ways either by adding or removing data and creating synthetic instances in the sample imbalance dataset at the time of dealing imbalance problem. To reveal the concealed pattern or the inner meaning of a data set is the main task of data mining [3].

The most popular sampling techniques are under-sampling and over-sampling. The under-sampling method randomly deducts majority class instances from the dataset, while the over-sampling method artificially generates and adds minority class instances to the dataset in order to balance the imbalanced dataset [4]. This can be attained by two procedures such as under-sampling and over-sampling or sometimes by blending over-sampling and under-sampling methods as a hybrid approach [5]. Although both methods are well-known and function skilfully at the time of classification. However, both have some significant problem which impacts the results. In an under-sampling method, some significant instances may be eliminated and an overfitting of data may occur in an over-sampling method [6]. To overcome this problem various methods have been proposed over the years, for example, SMOTE [4], AdaBoost [7], etc.

On the other hand, classification results are most often unstable in cost-sensitive learning as it is very difficult to get the accurate misclassification cost. Also, an ensemble classifier increases the classification accuracy of a single classifier, but ensemble learning does not solve the class imbalance problem either. The ensemble methods use sampling techniques to obtain balanced data in each iteration, as a result, they might suffer from overfitting or drop some potential information [5]. Moreover, Organizations face a great loss as some fraudulent failed to detect immediately. To solve the issue a method has been evolved by combining both the data mining and process mining [8] .

To deal with imbalanced datasets ensemble methods

Manuscript received July 11, 2018; revised December 07, 2018.
Md Yasir Arafat is with Wipro Limited as a Technical Lead;
E-mail: arafatix@gmail.com Web: http://arafat.info
Sabera Hoque was with Computer Science & Engineering Department, United International University, Bangladesh; E-mail: tumpa25@gmail.com
Shuxiang Xu is with School of Technology, Environments and Design as a lecturer, University of Tasmania, Australia;
E-mail: shuxiang.xu@utas.edu.au
Dewan Farid is from United International University as a Associate Professor, Bangladesh; E-mail: dewanfarid@cse.uiu.ac.bd

are needed to be designed specifically. In this paper, we study and review 12 states of the art techniques: SMOTE, AdaBoost, RUSBoost, EUSBoost, SMOTEBoost, MSMOTEBoost, DataBoost, Easy Ensemble, BalanceCascade, OverBagging, UnderBagging, SMOTEBagging for mining imbalanced data and compare their performance on 27 imbalanced datasets. In the rest of the paper has been arranged as follows. In section II we have discussed the different performance evaluator and their function. Section III presents different types of data balancing methods. Section IV presents the discussion on 12 imbalanced class classification algorithms. In section V we illustrate the characteristics of the used real-world datasets that have been collected from KEEL [9] data set repository. Section VI represents most significant outcome of our research. We have introduced some tables and figures to provide better insight of our experiments. The input parameters that has been used in our experiments is given in table III. In table VI, we have shown AUROC comparison. In figure 2 we have represented graphical view of table VI. Apart from that, we have introduced another useful process that is G-mean in table VII to compare algorithms. In figure 3 represents the line charts of table VII. To prove the significance of the research, we have included some other important methods such as F-measure and the resulting value is mentioned in table VIII with graphical representation in figure 4. Different execution or processing time that is mentioned in table IX with line charts in figure 5. Last but not least, the most crucial part of this section is the independent AUCROC line charts of algorithms and datasets (Randomly chosen 4 out of 27 used dataset: PIMA, ecoli2, ecoli3, glass01) shows in figure 6. Finally, we conclude this analysis in Section VIII.

II. The classifier performance evaluators

The performance of machine learning algorithms is evaluated by a standard evaluation matrix which is called confusion matrix, as illustrated in Figure 1 (for a 2 class problem). Here the columns are the predicted class and the rows are the actual class. In the confusion matrix, TN is the number of negative examples correctly classified (True Negatives), FP is the number of negative examples incorrectly classified as positive (False Positives), FN is the number of positive examples incorrectly classified as negative (False Negatives)and TP is the number of positive examples correctly classified.

TABLE I: 2x2 Confusion Matrix

Confusion matrix		
	Positive Prediction	Negative Prediction
Positive class	True Positive(TP)	False Negative(FN)
Negative class	False Positive(FP)	True Negative(TN)

The most common performance evaluation metrics related to imbalanced classes are: 1) Accuracy 2)Recall (sensitivity) 3) Specificity 4) Precision 5) F-measure and 6) Geometric mean (g-mean)[4]. Accuracy is the ratio between true decisions predict by a classifier. Sensitivity (also called True Positive Rate) and specificity (also called True Negative Rate) are used to monitor the classification performance of each individual classifier and recognise positive and negative examples respectively. [10]. Precision is used in problems interested in high performance in only one class and it is the ratio between true positive examples. F- measure and G-mean are harmonic averages and geometric averages of sensitivity and precision respectively. The geometric average of sensitivity and specificity is known as G-mean 2 [11].

ROC curve introduces the true positive rate (TPR) along the y-axis and false positive rate (FPR) along the x-axis. The classifier which produces accurate result would have an area under the curve (AUROC) of 1, where TPR = 1 and FPR = 0. At various cut - off points, the curve displays the TPR and FPR of the classifier at a different range.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

$$TPRate(Sensitivity) = \frac{TP}{TP + FN} \tag{2}$$

$$FPRate(specificity) = \frac{FP}{FP + TN} \tag{3}$$

$$precision = \frac{TP}{TP + FP} \tag{4}$$

$$G - mean = \sqrt{TPrate \times TNrate} \tag{5}$$

$$G - mean = \sqrt{sensitivity \times precision} \tag{6}$$

$$G - mean2 = \sqrt{TPrate \times TNrate} \tag{7}$$

$$G - mean2 = \sqrt{sensitivity \times specificity} \tag{8}$$

$$F - measure = \frac{2(precision.Sensitivity)}{precision + Sensitivity} \tag{9}$$

$$AUROC = \frac{TPrate + TNrate}{2} \tag{10}$$

The F1 score (also F-score or F-measure) is a weighted average evaluator that measure accuracy of a classifiers. This is a regular performance evaluation metrics interpreted as a better choice that combines precision and recall into a single value, by giving equal weight on both class [12]. The geometric average (G-mean) has been used by several researchers for evaluating classifiers on imbalanced datasets especially when performance of both classes is concerned and expected to be high at the same time. These measures point out the stability between classification performance on the majority and minority class. Both G-mean and F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0 [13]. In the case of imbalanced classification issues, insistence is established notably on the predictive accuracy of minority/positive class while having accuracy for the majority/negative class. This would correlate to high TPR and low FPR, and thus reflected by a high AUROC.

III. Data Balancing Methods

A. Sampling Technique

The customary evaluation procedures were unable to measure the exact model performance mostly when working

with real-life imbalanced datasets. Sampling techniques solve the class imbalanced classification problem with the redistribution of the imbalanced datasets in order to achieve an almost equal number of instances in two ways, which are called undersampling and oversampling methods. Sometimes a performance can be elevated by combining both the under and oversampling method which can be identified as hybrid approaches such as SMOTE with TOMEK or ENN method [5].Various intellectual algorithms have been proposed related to sampling technique [14], like SMOTE, borderline SMOTE, and Wilson's editing [15]. Hulse [16] observes the performance of diverse data sampling methods (around seven) using learning algorithms including investigational data sets, by discovering that the two most successful sampling algorithms are RUS and SMOTE.

1) Over-Sampling: The over-sampling method works with minority class instances by creating synthetic instances which causes no lose of any potential information. It can be done in two ways: (1) Random Over-sampling, and (2) Informative Over-sampling. Random over-sampling methods balance the datasets by randomly over-sampling the minority class instances.Whereas, informative over-sampling synthetically generates and adds the minority class instances into the dataset. The disadvantage of this method is overfitting of datasets as it adds replicated data from the original dataset. Some well-known over-sampling Methods are:

SMOTE (Synthetic Minority Over-sampling Technique) - this sampling method combines under-sampling of the majority class instances with over-sampling of the minority class instances [4]. Inspired by the favorable outcome of some popular concepts of using artificial instances like SMOTE [4], SMOTEBoost [4], and DataBoost [17], this theory proposes an adaptive method called ADASYN (Adaptive Synthetic Sampling Approach for Imbalanced Learning) [18]. With Borderline-SMOTE-I, to attain excellent prediction, the borderline instances of the minority class are over-sampled only. This sampling concept is different from the existing over-sampling ideas where the negligible instances or the arbitrary subset of the minority class are over-sampled [19] [20] [4]. Apart from that, We will mention another significant method called borderline Over-sampling in the Feature Space (BOSFS), that manage over-sampling method by creating novel synthetic minority instances with the existing borderline instances. The SVM classifier achieves higher recognition performance with this BOSFS method using the Euclidean distance, especially for the minority class instances [21].

2) Under-Sampling: The under-sampling method works with majority class instances, which removes instances randomly from a majority class to make the dataset balanced. It is best to use when the dataset is too big. Under-sampling methods are of two types: (1) Random under-sampling, and (2) Informative under-sampling. Random under-sampling method randomly chooses instances from the majority class, which later eliminates when the dataset gets balanced. Removing instances randomly may causes loss important information. On the other hand, informative under-sampling uses a pre-specified selection criterion to remove the majority class instances to balance the dataset [22].

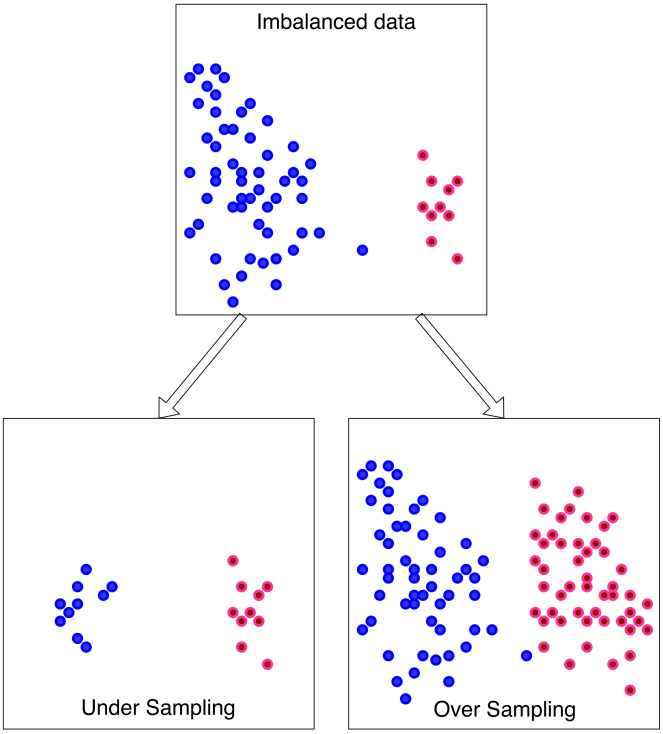

Fig. 1: Sampling Techniques.

B. Cost-sensitive Learning Method

The cost-sensitive learning (CSL) takes the misclassification costs into account in the learning process to minimize total cost. It is also a common approach to solve the class imbalanced problem [23]. The objective of the cost-sensitive learning is to achieve high accuracy to classify minority class instances. However, it is difficult to find the misclassification cost in different iterations as it depends on different types of errors.

C. Ensemble Method

Ensemble learning is the process of combining multiple classifiers to form a strong classifier to classify new instances with high prediction accuracy. Examples are RandomForest, Bagging, and Boosting. While constructing ensemble methods the most challenging task to be faced by the classifiers are: (1) the coalescence of the classifiers to be used, (2) the base classifiers used for ensemble must be naive to overcome overfitting, and (3) the base learners used should be as accurate as possible, and as distinct as possible.

IV. Imbalanced Class Classification Methods

In this section, we discuss the following machine learning algorithms: SMOTE, AdaBoost, RUSBoost, EUSBoost, SMOTEBoost, MSMOTEBoost, DataBoost, Easy Ensemble, BalanceCascade, OverBagging, UnderBagging, SmoteBagging that are widely used in many real-world imbalanced data classification applications [24].

1) SMOTE: This sampling method combines under-sampling of the majority class with a notable form of over-sampling of the minority class [4]. The concept of the method is developed with various investigational datasets, C4.5 as the base classifier, and Naive Bayes classifier including the AUROC curve [25]. This is an over-sampling approach where not only the instances are over-sampled but also the minority class builds some synthetic samples. One of the well-established examples of this approach is the identification of the optical character [26], where the authors produced additional training data by performing particular operations on real data. To perturb the training data the authors used operations like rotation and skew. Assume that, the total adjacent instances of minority class is K. According to the quantities of total instances needed, adjacent instances among the K (where k=5) instances are selected at random. For example, if the required over-sampling amount is 400%, then only 4 are selected from the 5 adjacent instances and 1 instance is produced in each direction. Synthetic instances are created by taking the difference between the adjacent instances and featured instances. Now assume 2 random numbers between 0 and 1 and we multiply this difference by the random number. Then add the resulting value to the feature vector. The minority class become more general when this method successfully forces the decision region. SMOTE algorithm which is the latest approach to over-sampling technique enhance the appropriateness of a classifier for a minority class. The combination of SMOTE and under-sampling method performs better than ordinary under-sampling technique. The novel approaches with SMOTE are examined using different datasets, with varying levels of imbalance. Merging SMOTE and under-sampling technique also functions better, based on possession in the ROC space. As SMOTE makes the setting territory larger and unspecified, it makes the classifier very much unique than that of other classifiers.

2) AdaBoost: AdaBoost is the most popular and effective ensemble classifier, with key functions to improve the performance of weak learners [27], [7]. During each of its iteration, instance weights are changed to see how the classification rate changes with focuses on misclassified instances in the next iteration. AdaBoost uses the weighted vote strategy to classify the unlabeled instances. In the class imbalanced situation, the AdaBoost algorithm performs efficiently as each of its iterations. Minority class instances are misclassified and given higher weights to consider in subsequent iterations. AdaBoost is a most desired and used boosting algorithm, which assumes a sequence of classifiers and incorporates the vote of each isolate classifier for classifying an unknown or known instances [28]. The algorithm performs efficiently with each of its iterations, the misclassified minority class instances are given higher weights. During every iteration of this algorithm, an infirm presumption is made by the foundation learner of the classifier. If the instance accurately classified its weight will be reduced and if failed to classify then the opposite will be done which means the weight will be increased. The significant characteristic of AdaBoost is that the approach mainly focuses on troublesome data points which have been misclassified by the weak classifier. The weak learner need no advanced knowledge and an optimally weighted majority vote uses by the classifier.

3) RUSBoost: RUSBoost is an under-sampling approach, which randomly removes the majority class instances to make the dataset balanced [29], [19]. It is an ensemble classifier as it combines the random under-sampling technique with a boosting approach (AdaBoost.M2 algorithm). It creates a balanced dataset using a random under-sampling technique in each boosting iteration and considers the majority-voting technique to classify new instances. RUSBoost is an ensemble classifier which is an extended version of SMOTEBoost that produces a quicker and easier approach with an achievement which is normally perfect and sometimes more than that of SMOTEBoost. Seiffert et al. [29] say that RUSBoost is an unprecedented mongrel data sampling technique, which has a strong outline to enhance the performance of models. The accuracy of RUSBoost is measured with SMOTEBoost [19]. The common thing among the classifiers is that these two classifiers RUSBoost and SMOTEBoost introduce data sampling with the popular AdaBoost classifier. RUS, a sampling technique that arbitrarily dispels instances from the leading class [30] used by RUSBoost. On the other hand, SMOTEBoost [4] which constructs new minority class instances applies the SMOTE technique. The uses of RUS technique into the boosting process are its intelligibility, fastness, and performance. On the other side, SMOTE is a puzzled and extended data sampling technique which makes the SMOTEBoost more complex and suffers for disadvantages of prolonged model training time as compared to RUSBoost. The main advantage of RUS classifier is that it reduces the period necessary for constructing a model, especially when constructing an ensemble of models. Though there is no universally conventional optimal class distribution, a balanced (50: 50) distribution is often thought to be near optimal [20]. Moreover, when instances of the minority class are extremely uncommon, a ratio closer to 35: 65 (minority: majority) may result in better classification performance.

4) EUSBoost: Enhancing ensembles for extremely imbalanced datasets by the evolutionary under-sampling method is called as EUSBoost [31]. It is based on the RUSBoost method as it combines random under-sampling with boosting techniques [5], [32]. EUSBoost starts with the under-sampling technique that continued until the currently finest under-sampled uplifted. This method is computationally more expensive compared to the RUSBoost algorithm as it executes EUS in all iterations of boosting. In real-world applications, the usefulness and aptness of EUS have been proved successfully [33]. But the main drawback of EUSBoost is that it seeks perfect base learners that causes diversity loss in the final output. EUSBoost training phase is computationally more expensive than other methods. This is due to EUS, which is executed in all iterations of Boosting.

5) SMOTEBoost: SMOTEBoost combines the oversampling technique SMOTE with the AdaBoost algorithm, which outperforms on both the SMOTE and AdaBoost algorithms for learning from imbalanced data [34]. It acquires a balanced

dataset by applying SMOTE in each round of iteration of the boosting process [4].

This is an extremely potential sampling/boosting method to learn from imbalanced data that outperform on both for SMOTE and AdaBoost algorithms. Before building the weak hypothesis during each round of iteration, a more balanced training data set SMOTE is applied to the training data. Hence, the algorithm for SMOTEBoost is analogous to that of RUSBoost. At this point, the application of SMOTE has two defects that RUSBoost is designed to overcome. First, it enhances the complexity of the algorithm. SMOTE must find the k nearest neighbors of the minority class examples and have to make an educated guess between them to make new instances. RUS, contrariwise, simply eliminate the majority class instances indiscriminately. Secondly, SMOTE requires an elongated time to train the model, as it is an oversampling technique. When uses larger training datasets, many models require to build for the accurate output which produces elongated training time. Whereas, shorter model training times will be required for smaller training data sets [4]. However, this causes the possibility of creating redundancy. If only the borderline and noisy majority instances are removed then the selection procedures try to find a representative subset of the majority samples. As a result, the complexity of the algorithm is increased and the training time of the model have prolonged as well, which makes the algorithm more complex than RUS.

6) MSMOTEBoost: MSMOTEBoost combines MSMOTE (Modified Synthetic Minority Over-sampling Technique) and AdaBoost algorithm [7]. MSMOTE is the modified and improved form of SMOTE method. MSMOTE categorizes the minority class instances into three types depending on the distances between the instances: border instances, security instances and noisy instances [35]. By using the nearest neighbors method MSMOTE algorithm generates synthetic instances. The classification of MSMOTEBoost is more accurate than SMOTEBoost for classifying minority class instances [36]. MSMOTE generates synthetic samples by calculating the space between each minority samples using the nearest neighbor technique. MSMOTE Technique has maintained the following strategy: firstly, the algorithm arbitrarily selects a data point from the nearest neighbor instances for the security instances. After that, the nearest neighbor for the border instances is selected and doesn't perform anything for latent noise instances. It is mandatory for this method to compute the space among the instances of a negligible class and all the instances in order to judge the samples type. Security instances can improve the function of the classifier. Whereas, Noisy instances reduce the performance of a model and border instances are very hard to classify for the model. The experimental analysis has proved that the prediction accuracy of MSMOTE model is outperformed by SMOTEBoost in case of negligible class [36].

7) DataBoost: The DataBoost-IM is an ensemble classifier that amalgamates data creation and boosting processes to achieve high classification accuracy to classify both the majority and minority class instances without omitting the majority and minority classes [17]. It built with the following three phases. Step1, an equal weight is assigned to each training instances. Here, the original training set is applied to build the initial classifier. Step 2, difficult instances are identified and for each of these core instances, a set of artificial instances is created. Finally, in Step 3, the artificial majority and minority class instances are merged with the original training set to create a balanced dataset.

8) EasyEnsemble: EasyEnsemble is to build classifiers to lead the sampling process for subsequent classifiers and to further utilize the majority class instances disregarded by under-sampling [37]. It has the advantage of combining boosting and bagging methods with data balancing method. The core concept of EasyEnsemble is straightforward and in some direction, it is analogous to the balanced Random Forest algorithm [38]. The final output of this method is an individual ensemble, though it acts like an ensemble of ensembles. Diverse researches [39], [40], [41], [42] amalgamate various ensemble techniques to attain stronger generalization. MultiBoosting [40], [41] amalgamates boosting and bagging [28] through applying boosted ensembles as foundation learners. Different ensemble strategies are combined to release a better solution like Stochastic Gradient Boosting [43] and Cocktail Ensemble [42].

9) BalanceCascade: BalanceCascade uses an under-sampling process to combine weak classifiers [44]. In this method, the sequential reliance among classifiers is predominantly utilized for lessening the duplicate information in the majority class. It conducts instance space for the under-sampling method to pursue proper knowledge [23]. To be successful in the rapid testing speed test usually the balanced cascade classifier is an appropriate choice [37]. BalanceCascade omits the duplicate instances in the majority class and confines instance difference to pursue convenient knowledge. Both BalanceCascade and EasyEnsemble are analogous to their structures. The most popular technique is called stacking, [44] which combines weak classifiers with EasyEnsemble and BalanceCascade.

EasyEnsemble and BalanceCascade both share a common strategy where they intentionally enhance the weights of the instances by adding higher misclassification cost at the time of processing a boosting approach. Both classifiers are designed to utilize the majority class instances which are mostly ignored by under-sampling in keeping the pace of training speed with high quality. They almost perform alike by sampling multiple subsets of the majority class. Training each of the subsets and finally combining all weak classifiers as an output. As ensemble subsets comprise more information than a single one, which makes both algorithms much better uses of the majority class instead of under-sampling. The same training times and the same number of weak classifiers are used by both the classifiers. The principal distinction is that BalanceCascade works with trained classifiers for subsequent classifiers. On the other hand, EasyEnsemble samples work with independent subsets.

10) OverBagging: OverBagging is the ensemble process of combining over-sampling with bagging methods. It randomly over-samples minority classes in each bagging iteration. It uses majority-voting technique to classify new

instances, as each classifier gives its decision and final classification made by a majority of votes. If a tie appears, then the class with minor instances are returned [36]. The entire over-bagging method can be done in 3 steps from its training phase to testing phase i.e. i) re-sampling, ii) constructing ensemble and iii) voting. Because of multiple minorities and majority class, it is very complicated to choose the re-sampling rate. The output of the algorithm shows that a variety of instances dominates recall value notably. The larger diversity consequences excellent recall for a minority whereas poor recall for majority classes.

11) UnderBagging: UnderBagging method uses a random under-sampling technique of majority class instances with bagging to build an ensemble classifier [14]. It also considers majority voting of classifiers to classify a new/ known instance. We may lose some valuable informative majority class instances when randomly sampling majority class instances. The core idea of this method is to educate each of the single components of the ensemble classifier with a balanced learning instance. By replacing an individual classification model, (Here the 1-NN rule) with an imbalanced training set, by an amalgamation of various classifiers, each model uses a balanced training set for its learning process. To earn this, as many training sub-samples as possible to get balanced subsets are beget. The number of sub-samples will be discovered by the difference between the number of prototypes from the majority class and that of the minority class. This workflow makes it possible to properly handle the difficulties of the imbalance and stays away from the implicit disadvantages to both the over and under-sampling techniques.

12) SMOTEBagging: SMOTEBagging builds a model by employing SMOTE with bagging methods to balance the class distribution. It over-samples from the minority class instances using SMOTE. In SMOTEBagging, we need to set the size/ amount of over-sampling of minority class instances and also the nearest-neighbors of instances [36]. SMOTE-Bagging is differed from UnderBagging and OverBagging, by involving in subset creation of synthetic instances. As claimed by the SMOTE algorithm, at first the (k) nearest neighbors and the volume of over-sampling from minority class (N) should be set. Here N=100, 200, 300, 400 and 500 i.e K=5 nearest neighbours. The correlative class must be reviewed at the time of all minority class ordination. An example is illustrated below: assume that the minority class X consists of 10 instances and the majority class Y consists of 60 instances. To over-sample, both X and Y through the same N value are applied so that the two classes are inner-imbalanced. In SMOTEBagging, x% value is used to control the instances from each class that is used for propagating novel instances.

V. Dataset description

A. Imbalanced Datasets

In this paper, we have considered 27 datasets from KEEL dataset repository [9]. Table II and Table V gives an outline of the features of the selected datasets.

TABLE II: Datasets description.(All are real world data)

Sl No.	**Dataset**	**Features**	**Instances**	**Imbalanced Ratio**
1	glass1	9	214	1.82
2	ecoli0_vs_1	7	220	1.86
3	wisconsin	9	683	1.86
4	pima	8	768	1.87
5	iris0	4	150	2
6	glass0	9	214	2.06
7	yeast1	8	1484	2.46
8	haberman	3	306	2.78
9	vehicle2	18	846	2.88
10	vehicle1	18	846	2.9
11	vehicle3	18	846	2.99
12	glass0123_vs_456	9	214	3.2
13	vehicle0	18	846	1.82
14	ecoli1	7	336	1.82
15	new-thyroid1	5	215	1.82
16	new-thyroid2	5	215	1.82
17	ecoli2	7	336	1.82
18	segment0	19	2308	1.82
19	glass6	9	214	1.82
20	yeast3	8	1484	1.82
21	ecoli3	7	336	1.82
22	page-blocks0	10	5472	1.82
23	yeast-2_vs_4	8	514	9.08
24	glass-0-1-6_vs_2	9	192	10.29
25	vowel0	13	988	10.10
26	yeast-0-5-6-7-9_vs_4	8	528	9.35
27	ecoli-0-1-3-7_vs_2-6	7	281	39.15

B. Algorithm Parameters

We have used the following parameters for our experiments:

TABLE III: Input Parameters for the experiment

Sl No.	Parameters	Value	Sl No.	Parameters	Value
1.	pruned	TRUE	9	Classification rate by algorithm and fold	YES
2.	InstancesPerLeaf	2	10	Header size in previous table	2
3.	Number of classes	2	11	Data used in previous table	TEST-TRAIN
4.	Folds	5	12	imbalanced Measure	Area Under the ROCCurve (AUROC)
5.	K Value	3	13	imbalanced Measure	Area Under the ROCCurve (AUROC)
6.	Distance Function	Euclidean	14	OS used for this experiments:	Linux and Mac OS
7.	Train Method	NORESAMPLING	15	CPU speed for this experiment.	2.5 GHz
8.	dataformat	keel			

TABLE IV: Strengths and weaknesses of 12 algorithms.

Sl No.	Algorithm	Strengths	Weaknesses
1	SMOTE	The classifiers are very much unique and computationally simple [4].	It is a perplexed and prolonged procedure.
2	AdaBoost	Prediction accuracy improve in case of minority data set and resolve the multi class imbalanced data set problem [7]	Ignore overall performance of the classifier.
3	RUSBoost	Simpler, easier, faster and less complex algorithm. RUS has been operated skilfully and swiftly [45].	The major handicap of this model is that important data can be destroyed.
4	EUSBoost	Easier, quicker and outperformed in highly imbalanced datasets.	Computationally more expensive than that of the other methods [32].
5	SMOTEBoost	This method is a perfect selection when the training instance size is excessively huge [4].	The possibility of creating redundancy. Besides more complex and prolonged than RUS.
6	DataBoost	Produce high accuracies of both classes (minority and majority) [17].	The use of many classifiers makes them more complex and produces output that is very hard to analyze
7	MSMOTE Boost	This method has better prediction accuracy [46].	Time-consuming and perplexing task.
8	Easy Ensemble	Reduce information loss. Faster training speed of undersampling [4].	Examining only binary classification problems [37].
9	Balance Cascade	Reduces the rate of creating duplicate information. Most popular method is stacking [44] .	Performance standard reduces in multi class.
10	Over Bagging	Performed masterly both in the binary and multi-class data set	Performance standard degrades in minority class [36].
11	Under Bagging	This procedure makes it possible to properly handle the difficulties of both the over and undersampling techniques [14].	When the experimental data size is not large the performance may be deteriorated.
12	SMOTE Bagging	Increased accuracy level [14].	Biased in favor of the majority class on high-dimensional data

TABLE V: Datasets detail description (Cont.)

Sl No.	Name	Attributes	Class
1.	glass0, glass1, glass01-23_vs_456, glass6 and glass-0-1-6_vs_2.	RI, Na, Mg, Al, Si, K, Ca, Ba and Fe.	Positive and Negative.
2.	ecoli1, ecoli2, ecoli3, ecoli0_vs_1 and ecoli-0-1-3-7_vs_2-6.	Gvh, Lip, Chg, Aac, Alm1 and Alm2.	Positive and Negative.
3.	wisconsin	ClumpThickness, CellSize, CellShape, MarginalAdhesion, EpithelialSize, BareNuclei, BlandChromatin, NormalNucleoli and Mitoses.	Positive and Negative.
4.	pima	Preg, Plas, Pres, Skin, Insu, Mass, Pedi and Age.	Positive and Negative.
5.	iris0	SepalLength, SepalWidth, PetalLength and PetalWidth.	Positive and Negative.
6.	yeast1, yeast3, yeast-2_vs_4 and yeast-0-5-6-7-9_vs_4.	Mcg, Gvh, Alm, Mit, Erl, Pox, Vac and Nuc.	Positive and Negative.
7.	haberman	Age and Year.	Positive and Negative.
8.	vehicle1, vehicle2, vehicle3	Compactness, Circularity, Distance_circularity, Radius_ratio, Praxis_aspect_ratio, Max_length _aspect_ratio, Scatter_ratio, Elongatedness, Praxis_rectangular, Length_rectangular, Major _variance, Minor_variance, Gyration_radius, Major_skewness, Minor_skewness, Minor _kurtosis, Major_kurtosis and Hollows_ratio.	Positive and Negative.
9.	new-thyroid1 and new-thyroid2	T3resin, Thyroxin, Triiodothyronine, Thyroidstimulating and TSH_value.	Positive and Negative.
10.	segment 0	Region-centroid-col, Region-centroid-row, Region-pixel-count, Short-line-density-5, Short-line-density-2, Vedge-mean, Vegde-sd, Hedge-mean, Hedge-sd, Intensity-mean, Rawred-mean, Rawblue-mean, Rawgreen-mean, Exred-mean, Exblue-mean, Exgreen-mean, Value-mean, Saturatoin-mean and Hue-mean.	Positive and Negative.
11.	page-blocks0	Height, Lenght, Area, Eccen, P_black, P_and, Mean_tr, Blackpix, Blackand and Wb_trans.	Positive and Negative.
12.	Vowel 0	TT, SpeakerNumber, Sex, F0, F1, F2, F3, F4, F5, F6, F7, F8, F9.	Positive and Negative.

VI. Experimental Results

TABLE VI: The AUC comparison of SMOTE, AdaBoost, RUSBoost, SMOTEBoost, EUSBoost, DataBoost, MSMOTEBoost, Over-Bagging, EasyEnsemble, BalanceCascade, Under-Bagging, SMOTEBagging on 27 imbalanced datasets.

Dataset	SMOTE	Ada Boost	RUS Boost	SMOTE Boost	EUS Boost	Data Boost	MSMOTE Boost	Over Bagging	Easy Ensem-ble	Balance Cascade	Under Bagging	SMOTE Bag-ging
glass1	**0.992**	0.809	0.77	0.783	0.783	0.7	0.762	0.77	0.77	0.847	0.754	0.744
ecoli-0_vs_1	**1**	0.969	0.976	0.972	0.962	0.94	0.969	0.979	0.941	0.941	0.979	0.979
wisconsin	0.955	0.961	0.956	0.965	0.944	0.960	0.939	0.98	0.989	**0.99**	0.961	0.967
pima	0.731	0.689	0.739	0.729	0.7	0.711	0.734	0.712	0.718	**0.881**	0.76	0.749
iris0	**1**	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.96
glass0	**0.875**	0.823	0.852	0.823	0.845	0.835	0.828	0.837	0.817	0.817	0.841	0.816
yeast1	**0.769**	0.652	0.704	0.712	0.688	0.699	0.709	0.707	0.689	0.677	0.723	0.694
haberman	**0.741**	0.581	0.629	0.614	0.554	0.561	0.628	0.598	0.639	0.636	0.664	0.666
vehicle2	**0.968**	0.966	0.972	0.981	0.971	0.966	0.947	0.960	0.952	0.952	0.961	0.967
vehicle1	0.753	0.7	0.743	0.752	**0.79**	0.966	0.728	0.708	0.712	0.712	0.766	0.746
vehicle3	0.755	0.681	0.764	0.744	0.722	0.691	0.738	0.720	0.728	0.728	**0.791**	**0.791**
glass-012-3_vs_4-56	**0.953**	0.877	0.879	0.918	0.914	0.897	0.918	0.884	0.901	0.901	0.888	0.909
vehicle0	**0.959**	0.931	0.958	0.976	0.922	0.944	0.939	0.949	0.929	0.929	0.952	0.947
ecoli1	**0.961**	0.822	0.912	0.847	0.898	0.806	0.892	0.867	0.882	0.882	0.899	0.904
new-thyroid1	0.986	0.931	0.932	**0.988**	0.96	0.957	**0.988**	0.954	0.955	0.955	0.966	0.977
new-thyroid2	**0.972**	0.954	0.918	0.971	0.935	0.957	0.948	0.934	0.946	0.954	0.954	0.969
ecoli2	**0.964**	0.851	0.865	0.909	0.928	0.808	0.909	0.871	0.867	0.867	0.898	0.914
segment0	**0.997**	0.99	0.991	0.993	0.991	0.991	0.991	0.990	0.985	0.987	0.985	0.982
glass6	**0.986**	0.849	0.909	0.855	0.906	0.841	0.922	0.883	0.893	0.893	0.890	0.911
yeast3	**0.918**	0.840	0.916	0.887	0.885	0.897	0.914	0.897	0.907	0.907	0.932	0.934
ecoli3	**0.95**	0.716	0.871	0.858	0.881	0.812	0.856	0.743	0.801	0.841	0.880	0.870
page-blocks0	0.964	0.930	0.947	0.939	0.902	0.874	**0.989**	0.936	0.953	0.956	0.956	0.955
yeast-2_vs_4	0.91	0.889	0.965	0.86	0.905	0.866	0.848	0.861	0.886	0.886	**0.954**	0.882
glass01-6_vs_2	**0.782**	0.588	0.679	0.632	0.785	0.649	0.601	0.582	0.611	0.625	0.678	0.672
vowel0	**0.971**	0.97	0.957	0.969	0.9854	0.97	0.943	0.951	0.947	0.947	0.946	0.971
yeast-0567-9_vs_4	**0.817**	0.645	0.845	0.781	0.757	0.737	0.765	0.715	0.751	0.751	0.788	0.802
ecoli-013-7_vs_26	0.731	0.648	0.83	0.831	0.804	0.548	**0.844**	0.74	0.817	0.817	0.802	0.828

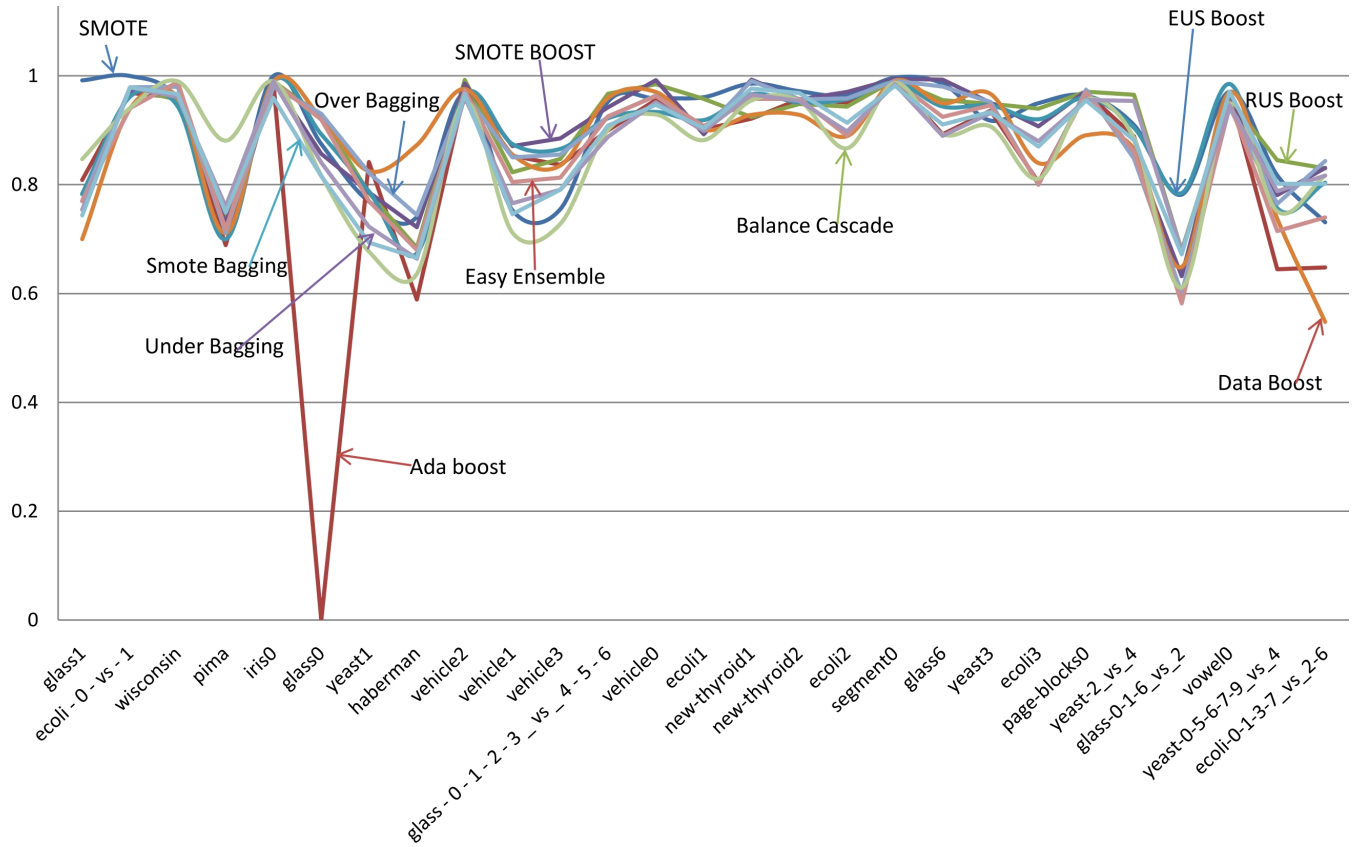

Fig. 2: The comparison among classifiers (AUROC value) for imbalanced data classification.

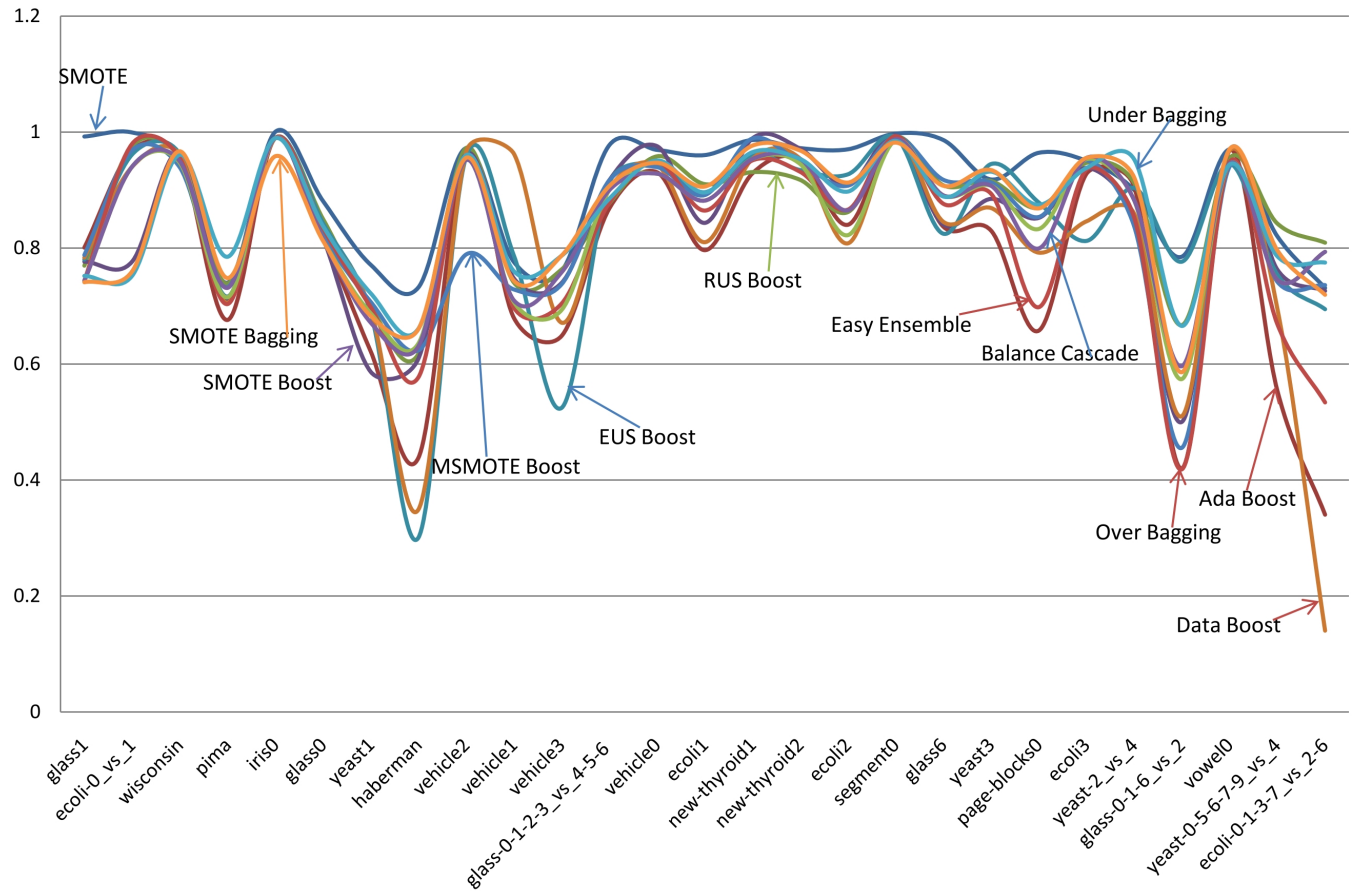

Fig. 3: The comparison among classifiers (G-mean value) for imbalanced data classification.

TABLE VII: The G-mean comparison of SMOTE, AdaBoost, RUSBoost, SMOTEBoost, EUSBoost, DataBoost, MSMOTE-Boost, Over-Bagging, EasyEnsemble, BalanceCascade, Under-Bagging, SMOTEBagging on 27 imbalanced datasets.

Dataset	SMOTE	Ada Boost	RUS Boost	SMOTE Boost	EUS Boost	Data Boost	MSMOTE Boost	Over Bag-ging	Easy Ensem-ble	Balance Cascade	Under Bagging	SMOTE Bag-ging
glass1	**0.992**	0.8006	0.77	0.784	0.788	0.794	0.787	0.743	0.743	0.743	0.751	0.740
ecoli-0_vs_1	**0.999**	0.976	0.976	0.7764	0.961	0.94	0.969	0.979	0.940	0.940	0.76	0.76
wisconsin	0.955	0.960	0.955	0.964	0.962	0.959	0.939	0.955	0.946	0.949	0.961	**0.967**
pima	0.74	0.675	0.739	0.711	0.716	0.705	0.732	0.703	0.715	**0.7312**	0.784	0.748
iris0	**1**	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.958
glass0	**0.881**	0.819	0.852	0.817	0.842	0.835	0.823	0.834	0.815	0.815	0.837	0.812
yeast1	**0.771**	0.621	0.698	0.586	0.688	0.693	0.707	0.698	0.687	0.671	0.721	0.682
haberman	**0.732**	0.438	0.617	0.609	0.300	0.3488	0.623	0.577	0.635	0.630	0.660	0.666
vehicle2	**0.972**	0.965	0.972	0.962	0.957	0.966	0.788	0.959	0.952	0.952	0.960	0.955
vehicle1	0.776	0.679	0.740	0.747	0.787	**0.962**	0.728	0.697	0.707	0.709	0.766	0.743
vehicle3	0.76	0.648	0.762	0.739	0.524	0.671	0.737	0.704	0.693	0.758	**0.786**	**0.7866**
glass-012-3_vs_4-56	**0.977**	0.868	0.876	0.916	0.912	0.901		0.916 0.878	0.899	0.899	0.885	0.906
vehicle0	**0.969**	0.931	0.958	0.975	0.944	0.943	0.939	0.949	0.928	0.928	0.951	0.947
ecoli1	**0.961**	0.796	0.912	0.843	0.896	0.81	0.891	0.865	0.882	0.882	0.895	0.905
new-thyroid1	0.986	0.928	0.930	**0.988**	0.959	0.954	**0.988**	0.951	0.953	0.953	0.965	0.976
new-thyroid2	**0.972**	0.952	0.917	0.971	0.934	0.954	0.947	0.931	0.945	0.954	0.954	0.968
ecoli2	**0.97**	0.84	0.861	0.907	0.926	808	0.908	0.866	0.822	0.865	0.897	0.912
segment0	**0.997**	0.989	0.991	0.993	0.991	0.991	0.991	0.993	0.985	0.987	0.985	0.982
glass6	**0.986**	0.836	0.908	0.841	0.824	0.845	**0.991**	**0.993**	0.985	0.987	0.984	0.981
yeast3	**0.918**	0.83	0.915	0.884	0.944	0.869	0.913	0.893	0.907	0.907	0.932	0.934
ecoli3	**0.95**	0.928	0.946	0.938	0.812	0.812	0.951	0.935	0.953	0.955	0.938	0.955
page-blocks0	**0.964**	0.657	0.869	0.851	0.878	0.792	**0.853**	0.698	0.832	0.799	0.874	0.868
yeast-2_vs_4	**0.9**	0.906	0.912	0.8528	0.9034	0.8592	0.8374	0.853	0.8824	0.8824	0.953	0.9214
glass-0-1-6_vs_2	**0.785**	0.4198	0.6668	0.5004	0.7762	0.51	0.4556	0.4184	0.574	0.5962	0.6654	0.5858
vowel0	**0.97**	0.968	0.9554	0.9692	0.9454	0.968	0.9404	0.9482	0.9454	0.9458	0.944	0.9704
yeast-0-5-6-7-9_vs_4	**0.82**	0.5538	0.842	0.7626	0.754	0.6954	0.7418	0.6688	0.7452	0.7452	0.7856	0.7958
ecoli-0-1-3-7_vs_2-6	**0.731**	0.34	0.809	0.726	0.6942	0.14	0.736	0.5338	0.7934	0.7934	0.7748	0.7192

TABLE VIII: The F-measure comparison of SMOTE, AdaBoost, RUSBoost, SMOTEBoost, EUSBoost, DataBoost, MSMOTEBoost, Over-Bagging, EasyEnsemble, BalanceCascade, Under-Bagging, SMOTEBagging on 27 imbalanced datasets.

Dataset	**SMOTE**	**Ada Boost**	**RUS Boost**	**SMOTE Boost**	**EUS Boost**	**Data Boost**	**MSMOTE Boost**	**Over Bag-ging**	**Easy Ensem-ble**	**Balance Cascade**	**Under Bagging**	**SMOTE Bag-ging**
glass1	**0.901**	0.753	0.706	0.722	0.723	0.729	0.733	0.687	0.681	0.681	0.691	0.677
ecoli-0_vs_1	**0.98**	0.954	0.967	0.722	0.944	0.94	0.955	0.973	0.909	0.909	0.691	0.717
wisconsin	0.89	0.945	0.939	0.950	0.948	0.947	0.919	0.939	0.931	**0.931**	0.941	0.952
pima	0.7	0.588	0.663	0.646	0.651	0.624	0.659	0.622	0.640	**0.642**	0.688	0.674
iris0	**1**	0.989	0.9896	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.956
glass0	**0.852**	0.630	0.785	0.752	0.780	0.802	0.749	0.776	0.740	0.740	0.771	0.743
yeast1	**0.722**	0.499	0.578	0.589	0.6	0.572	0.584	0.583	0.561	0.566	0.603	0.566
haberman	**0.699**	0.399	0.460	0.451	0.353	0.422	0.471	0.417	0.483	0.698	0.510	0.510
vehicle2	**0.955**	0.949	0.948	0.941	0.942	0.943	0.906	0.934	0.910	0.910	0.920	0.948
vehicle1	0.722	0.556	0.593	0.618	**0.645**	0.951	0.605	0.563	0.554	0.554	0.613	0.607
vehicle3	0.711	0.524	0.619	0.601	0.638	0.534	0.585	0.583	0.6006	0.57	**0.637**	**0.637**
glass-012-3_vs_4-56	**0.91**	0.835	0.805	0.872	0.847	0.867	0.872	0.840	0.829	0.829	0.808	0.849
vehicle0	**0.92**	0.892	0.906	0.946	0.866	0.906	0.872	0.911	0.842	0.842	0.876	0.883
ecoli1	**0.933**	0.750	0.806	0.755	0.799	0.833	0.802	0.780	0.778	0.668	0.777	0.787
new-thyroid1	0.988	0.884	0.847	**0.946**	0.874	0.951	**0.946**	0.937	0.862	0.862	0.899	0.946
new-thyroid2	**0.977**	0.940	0.810	0.922	0.833	0.951	0.915	0.898	0.863	0.899	0.896	0.911
ecoli2	**0.95**	0.765	0.699	0.814	0.778	0.722	0.814	0.773	0.697	0.655	0.755	0.778
segment0	**0.905**	0.986	0.966	0.984	0.99	0.984	0.981	0.984	0.940	0.951	0.939	0.969
glass6	**0.955**	0.774	0.749	0.772	0.782	0.755	0.859	0.790	0.689	0.689	0.693	0.809
yeast3	**0.85**	0.712	0.693	0.749	0.752	0.748	0.760	0.762	0.686	0.686	0.743	0.778
ecoli3	**0.82**	0.874	0.797	0.845	0.856	0.759	0.863	0.860	0.804	0.8008	0.845	0.839
page-blocks0	0.912	0.548	0.567	0.632	0.603	0.611	**0.631**	0.521	0.519	0.481	0.537	0.601
yeast-2_vs_4	0.81	0.815	0.713	0.706	0.671	0.754	0.679	0.709	0.662	0.662	0.716	0.720
glass-0-1-6_vs_2	**0.682**	0.333	0.273	0.404	0.383	0.488	0.317	0.317	0.206	0.219	0.270	0.404
vowel0	**0.771**	0.951	0.847	0.906	0.810	0.951	0.882	0.908	0.774	0.777	0.766	0.918
yeast-0-5-6-7-9_vs_4	**0.717**	0.370	0.536	0.542	0.405	0.505	0.515	0.477	0.368	0.368	0.421	0.468
ecoli-0-1-3-7_vs_2-6	**0.631**	0.75	0.141	0.444	0.388	0.283	0.741	0.869	0.128	0.128	0.194	0.45

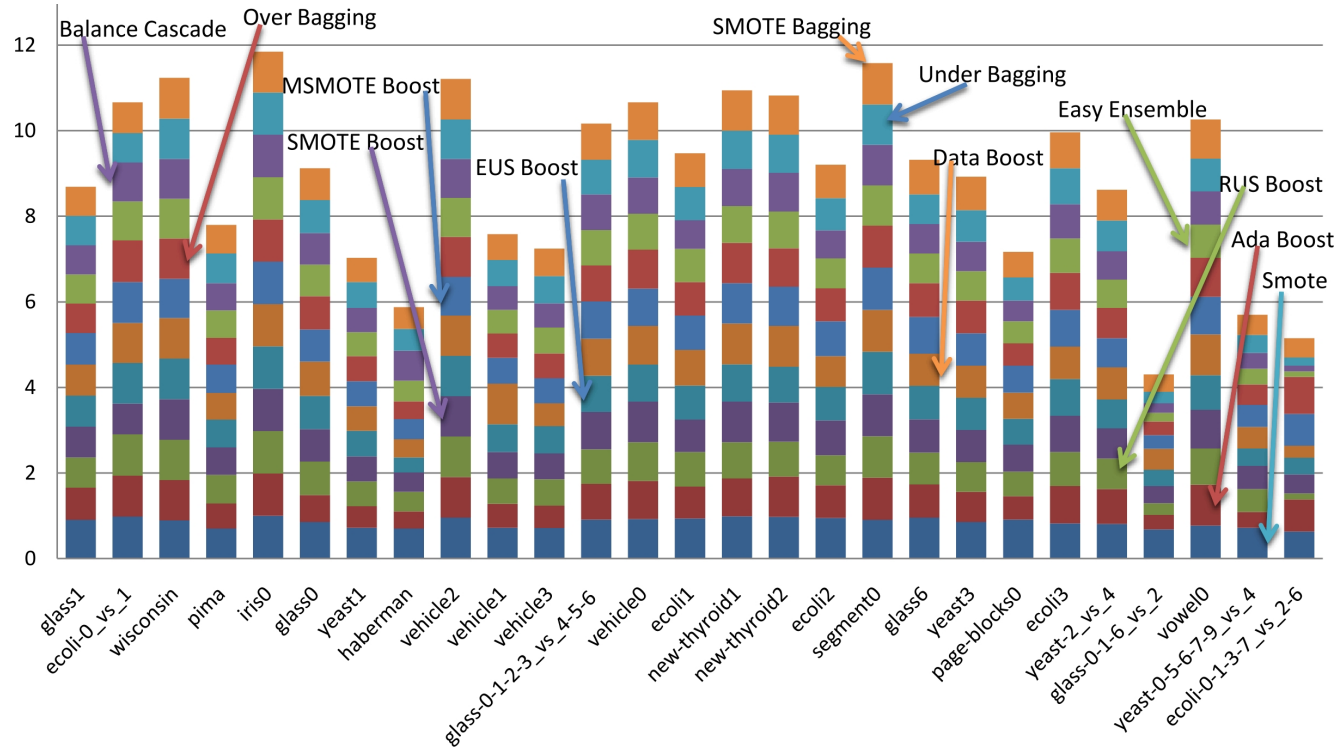

Fig. 4: The comparison among classifiers (F-measure) for imbalanced data classification.

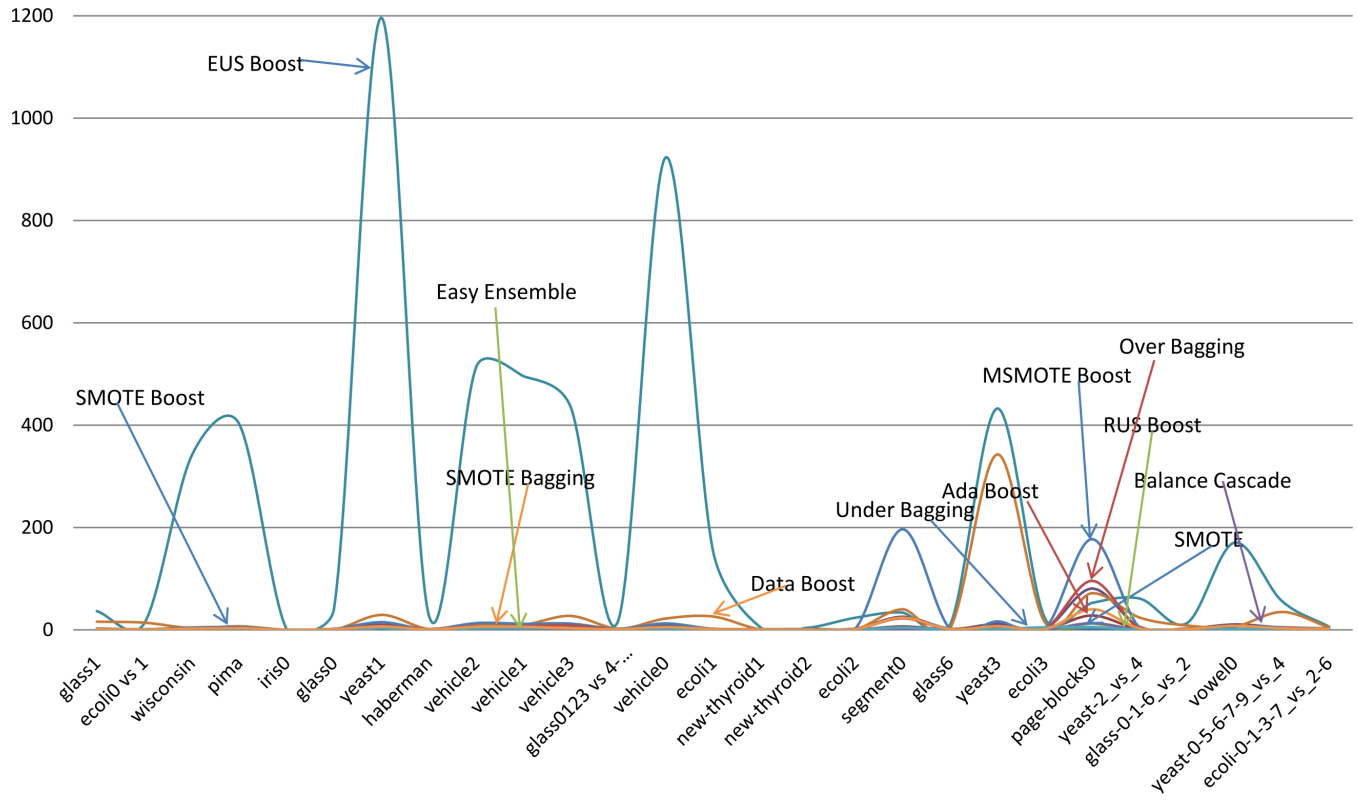

Fig. 5: The comparison among classifiers (Execution-time) for imbalanced data classification.

TABLE IX: The Processing speed i.e. the Execution Time (second) of SMOTE, AdaBoost, RUSBoost, SMOTEBoost, EUSBoost, DataBoost, MSMOTEBoost, Over-Bagging, EasyEnsemble, BalanceCascade, Under-Bagging, SMOTEBagging on 27 imbalanced datasets.

Dataset	SMOTE	Ada Boost	RUS Boost	SMOTE Boost	EUS Boost	Data Boost	MSMOTE Boost	Over Bag-ging	Easy Ensem-ble	Balance Cascade	Under Bagging	SMOTE Bag-ging
glass1	**0.02**	0.80	1.12	2.47	**36.57**	16.04	1.91	1.46	0.60	0.61	1.50	1.58
ecoli-0_vs_1	**0.14**	0.59	0.36	1.14	**12.76**	14.0	0.88	**0.39**	0.90	**0.43**	0.65	0.76
wisconsin	**0.07**	1.42	1.64	3.49	**5:40.51**	3.37	4.77	2.89	**0.94**	**1.16**	1.79	2.31
pima	**0.09**	1.75	2.40	4.84	**6:42.64**	7.18	5.41	3.47	1.42	**1.41**	2.16	3.32
iris0	**0.12**	**0.37**	0.25	0.25	1.88	0.25	0.25	0.24	0.24	0.27	**0.15**	0.24
glass0	**0.01**	0.78	0.81	1.83	40.7	**5:30.76**	1.84	1.447	0.65	0.61	0.94	1.28
yeast1	**0.23**	4.14	3.83	10.2	**19:55.82**	29.26	14.9	8.09	3.70	4.21	3.33	6.29
haberman	**0.02**	0.52	0.63	1.39	29.3	1.48	1.17	1.55	0.89	0.79	0.52	**0.96**
vehicle2	**0.19**	2.25	2.49	9.83	**8:33.50**	10.59	13.07	6.80	1.31	1.35	2.97	6.10
vehicle1	**0.16**	2.55	3.33	10.06	**8:15.48**	9.87	11.87	7.36	1.35	1.38	3.08	5.87
vehicle3	**0.19**	2.46	3.11	10.36	0.762	**27.2**	12.23	8.48	1.44	1.63	3.31	2.89
glass-012-3_vs_4-56	0.01	**0.49**	0.50	2.31	**17.85**	2.44	2.08	1.04	0.36	0.37	0.61	1.23
vehicle0	**0.14**	2.08	2.52	8.62	**15:22.81**	21.95	12.44	5.61	1.19	1.18	2.49	5.46
ecoli1	**0.02**	0.504	0.74	2.04	**2:34.58**		2.23	1.62	0.75	0.48	0.71	1.47
new-thyroid1	**0.01**	0.34	0.20	1.48	4.98	0.83	**1.39**	0.57	0.25	0.25	0.24	1.17
new-thyroid2	**0.01**	0.30	0.18	1.64	**4.08**	0.82	0.98	0.50	0.22	0.21	0.19	0.66
ecoli2	**0.01**	0.53	0.43	2.34	**23.6**		2.43	1.71	0.43	0.36	0.57	1.614
segment0	**0.58**	6.53	3.94	25.6		40.2	**3:16.28**	22.5	2.79	2.73	4.70	23.6
glass6	**0.01**	0.51	0.34	2.47	**9.34**	2.86	2.06	1.09	0.29	0.30	0.34	1.55
yeast3	**0.11**	2.33	1.54	11.6	**7:12.56**	**5:42.67**	16.4	7.78	1.06	1.01	1.43	7.17
ecoli3	**0.02**	0.65	0.52	2.48	**22.4**	14.7	2.54	1.82	0.39	0.37	5.22	1.69
page-blocks0	**1.53**	27.9	5.76	**1:20.76**		71.5	**2:57.31**	**1:35.86**	12.6	13.0	5.036	40.3
yeast-2_vs_4	**0.0346**	1:23:88	1:1.78	**5.9974**	1:1.0876	24.3274	4.639	2.6286	0.4326	0.5346	1:8.02	3.0108
glass-0-1-6_vs_2	**0.0088**	0.5302	0.339	3.14	**13.1314**	8.3452	2.0932	1.4462	0.2984	0.3358	0.5076	1.748
vowel0	**0.0938**	1.2344	1.2376	10.5064	**2:50.21**	6.1576	6.892	1.6404	1.4578	0.7336	1.2776	7.9642
yeast-0-5-6-7-9_vs_4	**0.0262**	1.591	0.7512	4.469	**56.5524**	34.8614	5.0538	3.4094	0.4764	0.6704	0.7926	3.1722
ecoli-0-1-3-7_vs_2-6	**0.01**	0.4786	0.1468	2.2808	**6.2428**	4.3724	2.3444	1.6222	0.185	0.2336	0.1646	1.7924

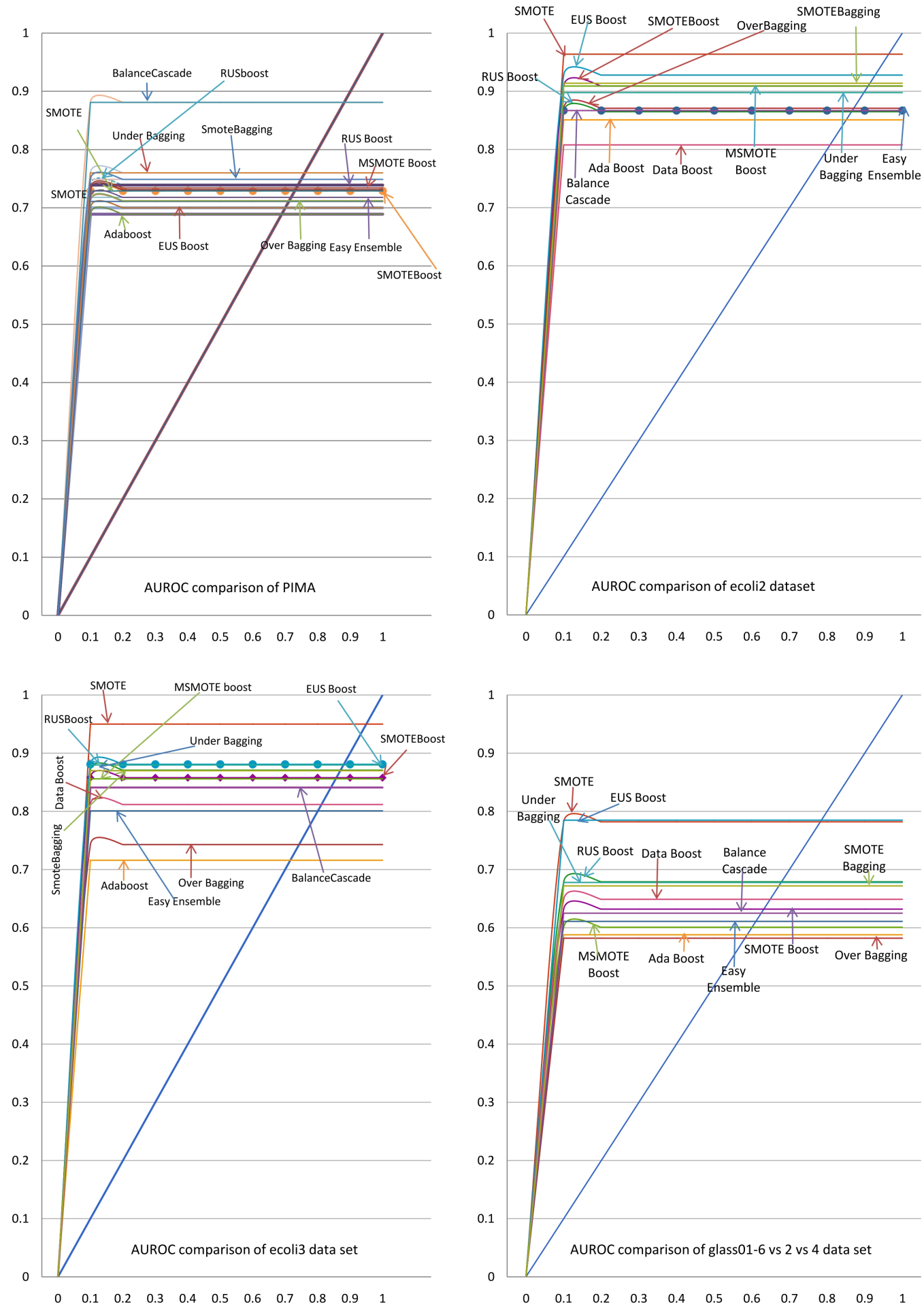

Fig. 6: AUCROC line charts of algorithms and datasets (Randomly given 4 out of 27 : PIMA, ecoli2, ecoli3, glass01).

VII. Results

We have tested the performance of 12 machine learning algorithms for imbalanced data classification using the Area Under Curve (AUC) with 5-fold cross-validation. The experimental results are tabulated in Table VI (in terms of ROC) and in Table IX (in terms of processing speed). Also in the table VII and VIII we present the experiment result of other two popular performance measure of imbalance data set such as G-mean and F-measure. Fig. 2, 3, 4 and 5 shows the comparison of the 12 classifiers using difference performance evaluator such as AUC, G-mean and F-measure and execution speed of the classifiers on 27 imbalanced datasets respectively. In all cases, we can see based on performance measures and processing speed of the classifiers in spite of being a single classifier SMOTE performs best compare with other ensemble classifiers.

VIII. Conclusion

In the last decade, several machine learning methods have been proposed for dealing with class imbalanced datasets to improve the performance of classifiers for classifying minority class instances. In this paper, we have reviewed some machine learning algorithms for imbalanced data classification. We have tested the performances of 12 imbalanced data classification methods: SMOTE, AdaBoost, RUSBoost, EUSBoost, SMOTEBoost, MSMOTEBoost, DataBoost, Easy Ensemble, BalanceCascade, OverBagging, UnderBagging, SMOTEBagging on 27 datasets from the KEEL Repository [9] with a high imbalanced ratio. The experimental results sum up that SMOTE has performed better in most datasets in comparison with ensemble classifiers. On the other hand, ensemble classifiers are more complicated to implement and understood. In general, a combination of both over-sampling and under-sampling techniques with ensemble classifier such as bagging and boosting achieve the highest accuracy for classifying both majority and minority class instances. In future work, we will apply these imbalanced data classification methods in real-life high-dimensional imbalanced big data and apply sampling techniques with RandomForest algorithm. In Table IV we represent the common strengths or weaknesses based on the experimental results of our mentioned algorithm with some differences.

References

[1] N. Tomašev and D. Mladenić, "Class imbalance and the curse of minority hubs," *Knowledge-Based Systems*, vol. 53, pp. 157–172, 2013.

[2] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[3] J. T. Lalis, "A new multiclass classification method for objects with geometric attributes using simple linear regression," *IAENG International Journal of Computer Science*, vol. 43, no. 2, pp. 198–203, 2016.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, no. 10, pp. 321–357, October 2002.

[5] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2012.

[6] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, and Y. Zhou, "A novel ensemble method for classifying imbalanced data," *Pattern Recognition*, vol. 48, no. 5, pp. 1623–1637, 2015.

[7] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[8] R. Sarno, R. D. Dewandono, T. Ahmad, M. F. Naufal, and F. Sinaga, "Hybrid association rule learning and process mining for fraud detection," *IAENG International Journal of Computer Science*, vol. 42, no. 2, pp. 59–72, 2015.

[9] 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*, vol. 17, no. 2-3, pp. 255–287, 2011.

[10] K. Veropoulos, C. Campbell, and N. Cristianini, "Controlling the sensitivity of support vector machines," *16th International Joint Conferences on Artificial Intelligence (IJCAI-99). Stockholm (Sweden, 1999)*, vol. 55, p. 60, 1999.

[11] R. P. Espíndola and N. F. F. Ebecken, "On extending f-measure and g-mean metrics to multi-class problems," *WIT Transactions on Information and Communication Technologies*, vol. 35, p. 10, 2005.

[12] L. A. Jeni, J. F. Cohn, and F. D. L. Torre, "Facing imbalanced data recommendations for the use of performance metrics," *Int Conf Affect Comput Intell Interact Workshops*, pp. 245–251, 2013.

[13] N. G. Hoang, A. Bouzerdoum, and S. L. Phung, "Learning pattern classification tasks with imbalanced data sets," *Pattern recognition*, pp. 193–208, 2009.

[14] R. Barandela, R. Valdovinos, and J. Sánchez, "New applications of ensembles of classifiers," *Pattern Analysis and Applications*, vol. 6, no. 3, pp. 245–256, 2003.

[15] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," *International Conference on Intelligent Computing*, vol. 3644, pp. 878–887, 2005.

[16] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Napolitano, "Rusboost: Improving classification performance when training data is skewed," *19th International Conference on Pattern Recognition*, pp. 1–4, 2008.

[17] H. Guo and H. L. Viktor, "Learning from imbalanced data sets with boosting and data generation: the databoost-im approach," *SIGKDD Explorations*, vol. 6, no. 1, pp. 30–39, 2004.

[18] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," *Proceedings of the International Joint Conference on Neural Networks*, pp. 1322–1328, 2008.

[19] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 1–6, June 2004.

[20] G. M. Weiss, "Mining with rarity: A unifying framework," *SIGKDD Explorations*, vol. 6, no. 1, pp. 7–19, 2004.

[21] K. Savetratanakaree, K. Sookhanaphibarn, S. Intakosum, and R. Thawonmas, "Borderline over-sampling in feature space for learning algorithms in imbalanced data environments," *IAENG International Journal of Computer Science*, vol. 43, no. 3, pp. 363–373, 2016.

[22] M. Y. Arafat, S. Hoque, and D. M. Farid, "Cluster-based under-sampling with random forest for multi-class imbalanced classification," *11'th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, pp. 1–6, 2017.

[23] K. M. Ting, "An instance weighting method to induce cost-sensitive trees," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 3, pp. 659–665, 2002.

[24] S. Hoque, M. Y. Arafat, and D. M. Farid, "Machine learning for mining imbalanced data," *International Conference on Emerging Technology in Data Mining and Information Security (IEMIS 2018)*, no. 1, p. 10, 2018.

[25] J. R. Quinlan, "Boosting, bagging, and c4.5," *Proc. 13th Nat'l Conf. Artificial Intelligence*, pp. 725–730, 1996.

[26] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, pp. 855–868, 2009.

[27] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, D. Haussler, R. E. Schapire, and M. K. Warmuth, "How to use expert advice," *Journal of the ACM (JACM)*, vol. 44, no. 3, pp. 427–485, 1997.

[28] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine Learning*, vol. 36, no. 1-2, pp. 105–139, 1999.

[29] T. M. Khoshgoftaar, C. Seiffert, J. V. Hulse, A. Napolitano, and A. Folleco, "Learning with limited minority class data," *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pp. 348–353, 2007.

[30] G. E. A. P. A. 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 - Special issue on learning from imbalanced datasets*, vol. 6, no. 1, pp. 20–29, 2004.

[31] S. García and F. Herrera, "Evolutionary undersampling for classification with imbalanced datasets: proposals and taxonomy," *Evolutionary Computation*, vol. 17, no. 3, pp. 275–306, 2009.

[32] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera, "Eusboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling," *Pattern Recognition*, vol. 46, no. 12, pp. 3460–3471, 2013.

[33] D. J. Drown, T. M. Khoshgoftaar, and N. Seliya, "Evolutionary sampling and software quality modeling of high-assurance systems," *IEEE Transactions on Systems,Man, and Cybernetics, Part A: Systems and Humans*, vol. 39, no. 5, pp. 1097–1107, 2009.

[34] K. S. WOODS, C. C. DOSS, K. W. BOWYER, J. L. SOLKA, C. E. PRIEBE, and J. W. PHILIP KEGELMEYER, "Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 6, pp. 1417–1436, 1993.

[35] M. Buckland and F. Gey, "The relationship between recall and precision, journal of the american society for information science," *Journal of the American Society for Information Science*, vol. 45, no. 1, pp. 12–19, 1994.

[36] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," *IEEE Symposium Series on Computational Intelligence and Data Mining*, vol. 19, no. 8, pp. 324–331, 2009.

[37] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory under sampling for class imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 539–550, 2009.

[38] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Information Sciences*, vol. 275, pp. 314–347, 2014.

[39] J. H. Friedman, "Stochastic gradient boosting," *Comput. Stat. Data Anal.*, vol. 38, no. 4, pp. 367–378, February 2002.

[40] G. I. Webb, "Multiboosting: A technique for combining boosting and wagging," *Machine Learning*, vol. 40, no. 2, pp. 159–196, 2000.

[41] G. I. Webb and Z. Zheng, "Multistrategy ensemble learning: Reducing error by combining ensemble learning techniques," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 8, pp. 980–991, 2004.

[42] Y. Yu, Z. H. Zhou, and K. M. Ting, "Cocktail ensemble for regression," *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pp. 721–726, 2007.

[43] J. H. Friedman, "Stochastic gradient boosting," *Comput. Stat. Data Anal*, vol. 38, no. 4, pp. 367–378, February 2002.

[44] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[45] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Napolitano, "Rusboost: A hybrid approach to alleviating class imbalance," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 1, pp. 185–197, 2010.

[46] S. Hu, Y. Liang, L. Ma, and Y. He, "Msmote: Improving classification performance when training data is imbalanced," *2nd International Workshop on Computer Science and Engineering(WCSE 2009)*, vol. 2, pp. 13–17, 2009.