Journal of Theoretical and Applied Information Technology
29th February 2020. Vol.98. No 04
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ENTROPY-BASED FUZZY RANDOM FOREST FOR IMBALANCED DATASETS

[1] SHROUK EL-AMIR, [2] SAMIRELMOUGY

[1]Department of Computer Science, Faculty of Computers and Information, Zagazig University, Zagazig, Sharkia, Egypt

[2] Department of Computer Science, Faculty of Computers and Information, Mansoura University, Mansoura, Egypt
E-mail: [1]s.shrouk64@gmail.com, [2]mougy@mans.edu.eg

ABSTRACT

With the recent wave of data analytics accessing every domain, there is a growing interest in handling an imbalanced classification problem. In many datasets, the positive class size is extra smaller compared with the major class (negative class), as in the case of disease detection, cyber-attacks, and many data mining applications. Among the different algorithms that addressed this problem, Random Forest (RF) attracted many researchers because of its general robustness. But, RSs and other cost-sensitive algorithms are suffering from low sensitivity and low precision according to positive class when dealing with imbalanced dataset problem. In this paper, we propose and develop an Entropy-based Fuzzy Random Forest (EFRF) algorithm to deal with imbalanced classification problem. Fuzzy membership is applied to the training instances such that different instances offer different contributions to the classifiers. Samples that have a higher class certainty are assigned to larger fuzzy memberships. EFRF uses the entropy to pay more attention to the samples with higher class certainty to result in more robust decision making to avoid losing information like other undersampling algorithms. The proposed algorithm showed promising results compared to other imbalanced classification techniques including Entropy-based Fuzzy Support Vector Machine (EFSVM) technique. It featured both high precision and high recall which makes it an excellent choice for security-wise application.

Keywords: *Imbalanced Dataset, Random Forest, Cost-Sensitive Learning, Sampling, Information Entropy*

1. INTRODUCTION

Many important real-world applications are characterized by highly imbalanced data including medical diagnosis [1], cyber-attacks detections [2], face recognition[3], credit card fraud detection [4], etc. The main challenge with this machine learning problem is the intrinsic tendency of classic classifiers to recommend the major class labels much over the minor class labels. However, some machine learning algorithms have been modified to handle the imbalanced classification problem as Random Forest (RF) [5], Adaptive Boosting (Adaboost) [6, 7], Gradient Boosting [8], and Entropy-based fuzzy support vector machine [9].

The different approaches to face the imbalanced data issue in the literature can be divided into three groups: data level, algorithm level and cost-sensitive approaches [10]. At data level approaches, data is modified by resampling the main training data space for better balancing of the two classes [11-13]. At algorithm level approaches, the algorithm is modified and adapted to handle the imbalanced data. At cost-sensitive learning solutions, the problem is addressed by assigning higher misclassification costs for instances belonging to the minority (positive) class [14].However, data-level approaches have some drawbacks caused by resampling. The best distribution of the training samples is sometimes unknown. Under-sampling may cause loss of the class information, while oversampling may cause overfitting. Algorithm level approaches also have some drawbacks such as sensitivity to noisy data and outliers, and sometimes harder to fit.

RF classifier is characterized by its noise resistance. It is known for its high performance and speed in both training and prediction stages. Using a standard top-down induction process, we can construct a number of unique decision trees. Each decision tree is constructed using a different sample of the training data recursively until it reaches to the maximum depth. Then, a pruning process is applied after the tree is completely constructed. The predicted class of a testing sample is calculated by aggregating the outcomes of the decision trees ensemble using majority voting. Adaboost [6, 7] is an ensemble algorithm that combines several weak learners into a stronger one. It is very simple to implement and suits any kind of classification problem. Gradient Boosting [8] is an ensemble algorithm that combines several decision trees into a stronger classifier sequentially. It gives a nice strategy to deal with unbalanced datasets.

Using Entropy-based Fuzzy Membership (EbFM) [9] yields ensuring the significance of the (minor) positive samples through high sensitivity and high precision. Support Vector Machine (SVM) tries to solve the imbalanced dataset problem. But its performance is not sufficient enough causes it ignores the difference between the negative and positive classes and deals with all samples as the same importance.

To solve the drawback of SVM, Fan et al. [9] proposed EFSVM approach which stratifies fuzzy membership to every input and recasts SVM. Its performance becomes more sufficient according to imbalanced dataset problem. The results show that the performance of RF is better than SVM concerning data size and mixed type of the features. But, RF suffers from the imbalanced dataset problem. In this work, we propose an Entropy-based Fuzzy Random Forest algorithm (EFRF) adopted from EFSVM.

The remainder of the research is formed as follows: Section 2 discusses the background. The EbFM estimation approach based on EFSVM algorithm [1] is discussed in Section 3. A detailed description of EFRF is presented. Section 4 summarizes the experimental design and shows and evaluating its results using real-world unbalanced examples. Section 5 summarizes and concludes this study. Section 6 presents the future work.

2. BACKGROUND

Because the imbalanced data problem has become a serious snag, researchers had been proposed various solutions to deal this it. This include: data level techniques [15], algorithm level techniques [16] and an integration of algorithm and data level techniques that are called as cost-sensitive techniques. Regarding of the data level techniques, the main training data is modified to obtain suitable data to develop the performance of the standard classification algorithms during facing the imbalanced data. There are two ways for applying the data level techniques: over-sampling approaches, and under-sampling approaches.

Over-sampling approaches [17] modify the samples to obtain balanced data by providing them with new positive instances. The simplest oversampling approaches are called Random Over-Sampling (ROS) [11] that copies positive instances from the original data set in a random way till the number of positive instances becomes close to the number of negative instances.one of the most public over-sampling approach Synthetic Minority Over-sampling Technique (SMOTE) that synthesis new minority instances between the original minority instances.

Under-sampling approaches delete various instances from the negative class to balance the dataset. The most popular under-sampling approach is Random Under-Sampling (RUS) [11] that deletes the negative instances from the main dataset in a random way to balance the data set. Regarding of the algorithm level techniques, the standard learning algorithms are adjusted to focus on a decision threshold biased towards the positive class.

Cost-sensitive techniques merge both of the data level and algorithm level techniques by giving higher misclassification costs to positive instances and reducing the cost in a comprehensive way. The misclassification costs are commonly represented by a cost matrix C in which C(i, j) indicates the costs of classifying a sample belonging class i to class j 18][. The popular Cost-sensitive methods are Gradientboost [8] and Adaboost (Adaptive Boosting) [6, 7], which allocate weight to the samples through training. In each iteration, weight of the misclassified samples is increasing while correctly classified is decreasing. Contained weights correction imposes on the learning process to be attending more on misclassified in subsequent iterations. In the case of imbalanced data, the minority class is incorrectly classified, and hence the boosting increases the accuracy of the results.

2.1 Random Forest

Random Forest (RF) [5] is a group of decision trees which is generally used in classification. Its structure is shown in Fig. 1 [19] while Fig. 2 shows the classification process [19]. RF is used in many works because of its high performance compared with other classification algorithms [20]. Breiman [19, 21, 22] suggested RF that includes a special randomness layer to bagging. In standard common trees, every node is slit by choosing the best split among all variables. This planning performs very well rather than many other classifiers, including SVM. Also, it is shown that it achieved good performance against overfitting.

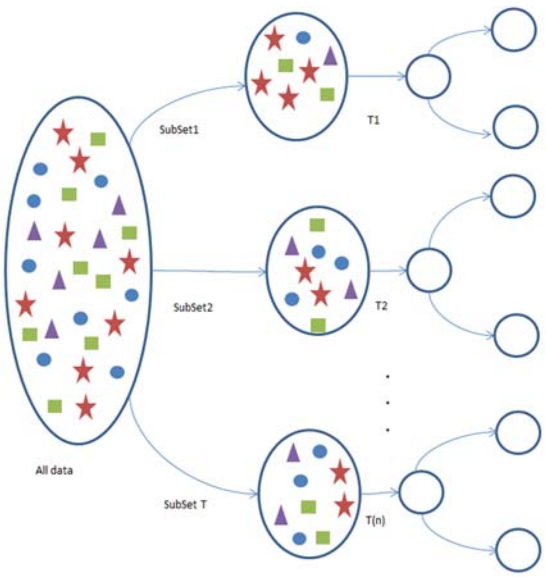

Fig.1 Random Forest structure.

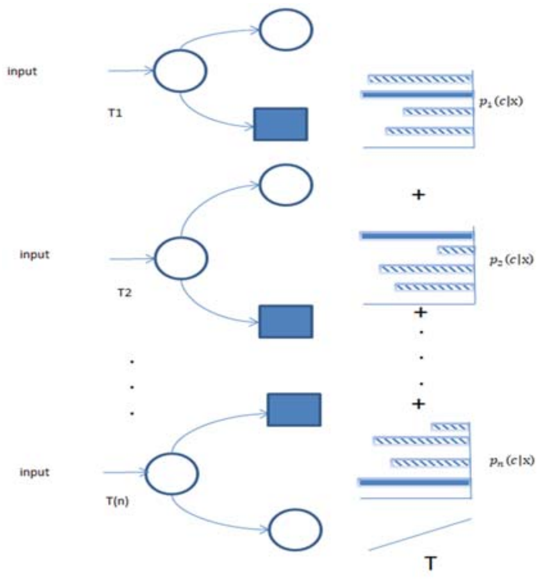

Fig.2 Random Forest classification process [19].

2.1.1 Training process

In RF training, bagging has been utilized to generate sample subsets by sampling in a random way that comes from the training data. One set of samples is utilized to structure one decision tree. At each slitting node n, sample set S_n is slit into S_l *and* S_r by matching the feature quantity value X_i with a value of threshold τ.The method of the splitting node selects specific collections that can divide the most samples from among selected features $\{f_k\}_{k=1}^{K}$and threshold $\{\tau_h\}_{h=1}^{H}$ in a random way for every class. The suitable number selections feature is the feature square root dimensionality. The estimation function applied to select the perfect collection is the information Gain, ΔG. The splitting process is repeated until either reaching its maximum depth or the information gain is equal to zero. Then, a leaf node is generated, and the probability of the class $P(c|l)$ is calculated.

2.1.2 Classification process

In regression, RF uses the average prediction of each tree, $p_t(c|\mathrm{x})$ and predicts with this instance using Eq. (1) [19].

$$P(c|\mathrm{x}) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|\mathrm{x}) \qquad (1)$$

In classification, a test instance is computed to get the predicted class by collecting the predictions of the decision trees set using majority voting, $p_t(c|x)$, according to Eq. (2) [19].

$$P(c|\mathrm{x}) = \arg Max_{Voting} p_t(c|x) \qquad (2)$$

The entropy has been utilized to gauge the class certainty. We use a fuzzy membership evaluation [9] in this paper to give the fuzzy membership of every instance depending on the class certainty of each sample. The certainty of class proves the certainty of the instances which has been classified to a corresponding class.

Because of the entropy has been a suitable approach for calculating the degree of certainty, we use it to evaluate each instance certainty. To achieve this, the fuzzy membership valuation using the entropy has been presented by locating the training samples fuzzy membership. With the help of the EbFM, our proposed EFRF algorithm has the ability for dealing with imbalanced datasets trouble. Because of the significance of the negative class is lower than the positive class in unbalanced datasets problem, the algorithm ought to make the classifier to be more attentive to the positive instances.

So, we assigned the large fuzzy memberships which are calculated by the EbFM evaluation, which is based on the standard which the instances with low-class certainty will easily mislead the decision surface, so we ought to make their importance weak during the learning stage.

2.2. Evaluation Metrics

The quality measures of classification have been structured from a confusion matrix that arranges every class samples with regard to their correct or incorrect identification. Eight metrics will be used which are known with unbalanced datasets problem. Geometric Mean (GM) [23, 24] is able to avert the problems unlike standard accuracy metric and is calculated as:

$$GM = \sqrt{sensitivity.specificity} \qquad (3)$$

where $sensitivity = \frac{TP}{TP+FN}$ and $specificity = \frac{TN}{FP+TN}$ This measurement seeks to make the precision of one of the two classes as large as possible.

Table1: Confusion matrix.

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

Another metric which is utilized in an unbalanced class snag is called β-f-measure (β-f-m) that is calculated as [25]:

$$\beta - \mathrm{f} - \mathrm{measure} = \frac{(1+\beta^2)(positive\ predictive\ value.\mathrm{sensitivity})}{(\beta^2.positive\ predictive\ value) + \mathrm{sensitivity}} \qquad (4)$$

where $positive\ predictive\ value = \frac{TP}{TP+FP}$.

The sensitivity and positive predictive value values have been commonly used as recall and precision, respectively. In these troubles, the β parameter value is equal to one assigning the same significance to the sensitivity and the positive predictive value. But, assigning β a value of one is not appropriate in an unbalanced snag during facing this class unbalance in the training and test samples together [26]. So, we need to use β with larger for increasing the sensitivity than in increasing the positive predictive value.

Kappa statistics: The value for finding the matching degree among intervals [0:1] is Cohen's kappa. The value of Cohen's kappa is characterized if [0:0.2], as weak, [0.2:0.4] as fair, [0.4:0.6] as moderate, [0.6:0.8] as good performance, and [0.8:1.0] as exemplary agreement [27].

While traditional accuracy is not a good method of evaluating the classifiers performance which deals with the imbalanced dataset, another alternative is balanced accuracy which is calculated as:

$$BalancedAccuracy = \frac{TP}{2(TP+FN)} + \frac{TN}{2(TN+FP)} \qquad (5)$$

Through computing the average of the percentage of positive class instances correctly classified and the percentage of negative class instances correctly classified28][.

MCC [29] is a measure that collects all the confusion matrix values, bearing in mind errors and correct classification in overall classes, as shown in Eq.(6).

$$MCC = \frac{TP \cdot TN + FP \cdot FN}{\sqrt{POS \cdot NEG \cdot PPOS \cdot PNEG}} \qquad (6)$$

Using the confusion matrix is not enough when dealing with the imbalanced data trouble. Consequently, researchers prefer to graphically

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

evaluate the performance through a set of cases by using the demonstrative valuation tools, like Receiver Operating Characteristic (ROC) curve.

ROC curve is a popular valuation method, taken from radio signal analysis [30]. ROC curve shows the tradeoff between FP rate and the TP rate. It will misclassify more negative instances as positive instances. It can be planned to use different probability threshold, from 0 until 1, for predicting positive instances. FP= $\frac{FP}{TN+FP}$ while the true positive rate and recall are the same thing. A ROC curve is considered to be efficient when the curve is approaching to the top left corner.

3. ENTROPY-BASED FUZZY RANDOM FOREST (EFRF)

Because the calculating of fuzzy membership is the main stage of FRF, we primarily present an EbFM estimation approach based on EFSVM algorithm [1]. Thereafter, our EFRF algorithm is proposed by adapting EbFM in which its architecture of EFRF is shown in Fig. 3

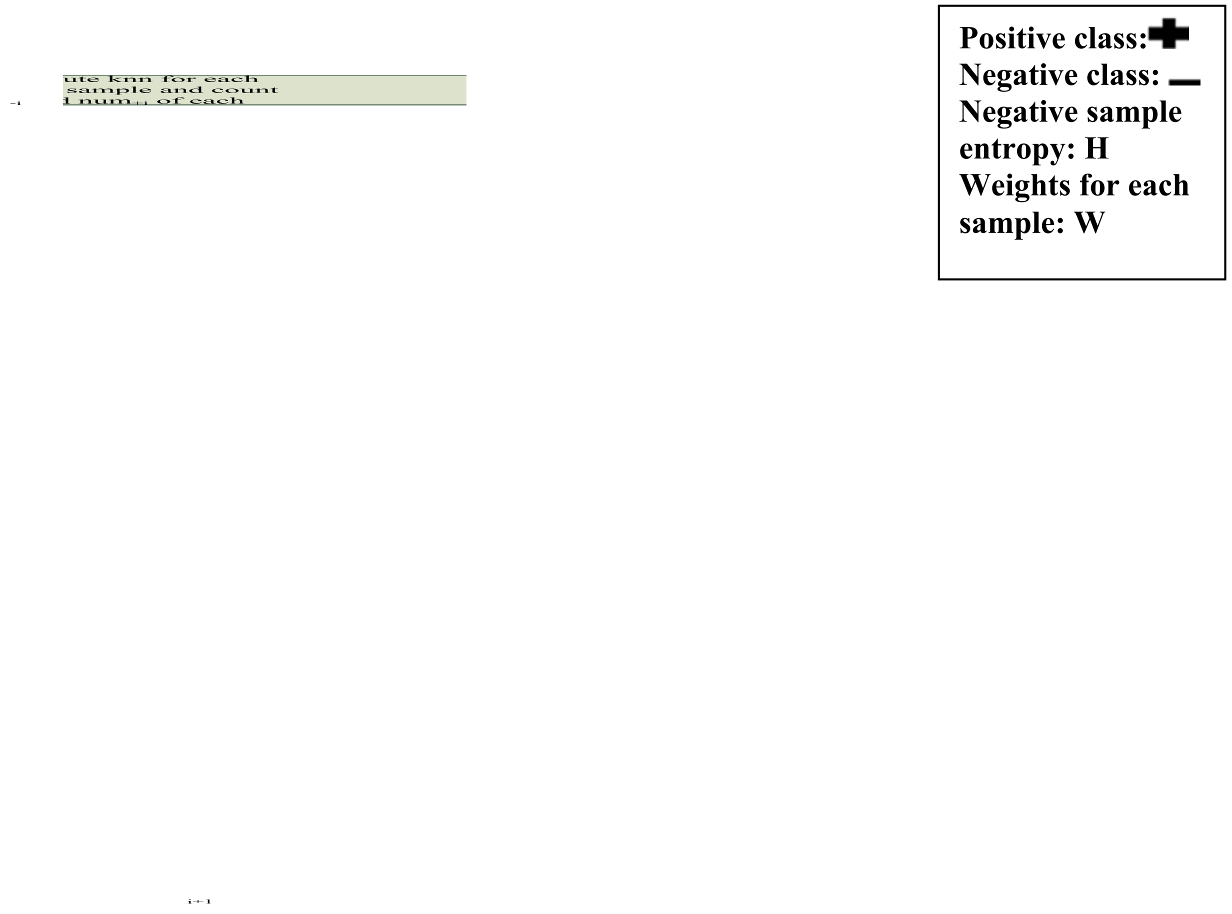

Fig. 3 Main architecture of EFRF

3.1. Entropy-Based Fuzzy Membership

In EbFM evaluation approach, a positive instances is assigned to a larger fuzzy membership because the minority class (positive class) is additional significant than the majority class (negative class) when facing unbalanced troubles. Here, we assign "1.0" value to positive instances for ensuring the positive class significance and fuzzify negative instances according to their class certainty. The training instances take the fuzzy membership according to their class certainty. So, entropy is used since it is the certainty about the information source. Suppose that the training samples $\{x_i, y_i\}_{i=1}^{N}$, $y_i \in \{+1, -1\}$, $y_i = +1$ refers to the sample x_i belongs to the positive class, else it belongs to the negative one. The x_i probabilities belonging to the positive and negative class are p_{+i} and p_{-i}, respectively. The entropy of x_i is defined as [9]:

$$H_i = -p_{+i} \ln(p_{+i}) - p_{-i} \ln(p_{-i}) \quad (7)$$

where p_{+i} refers to the sample probability belonging to positive class while p_{-i} refers to the sample probability belonging to negative class. The probability computation is depending on its knn. Figures 4 and 5 show how we calculate k nearest neighbors for each negative sample x_{-i}.. Suppose that there are two classes: one refers to positive class and the other to a negative class. Assume that x_1 is the negative sample which we want to calculate its nearest neighbors where neighbors, K=7. First, k closest instances to x_1 are determined through using Euclidean distance for calculating the distance between each negative and positive instances. Second, the seven instances are selected which have the smallest distance. So, these instances will be used for calculating the probabilities of x_{-i}.

Next stage is to select knn{ x_{i1},..., x_{iK} }; assume it is x_i as we mentioned above. The number of positive num_{+i} and negative instances num_{-i} are enumerated in these k chosen samples. Ultimately, the probabilities of x_i are calculated as [9]:

$$p_{+i} = \frac{num_{+i}}{k} \quad (8)$$

$$p_{-i} = \frac{num_{-i}}{k} \quad (9)$$

Using Eq. (7) [9], the negative samples entropy becomes H = { H_{-1},H_{-2},..., $H_{-n\_}$}, where n_ is the negative samples number, H_{max}, and H_{min}, refer to the maximum and minimum entropy respectively.

The negative samples are separated into subsets (m), with an increasing entropy order depends on their entropy as appeared in the negative samples algorithm separation [9]. So, H_{sub_1}, < H_{sub_2} <... < H_{sub_m}, where H_{sub_1} indicates the entropy of subset sub_1.

Fig. 4 shows calculating Euclidean Distance of X_1, while Fig. 5 illustrated detecting k nearest neighbors of X_1.

Algorithm 1:The Negative Samples Algorithm Separation

```
1    Begin
2    Input: m = number of subset m
3    Variables: m = number of subset m, $N\_$ = number of negative samples and
4                  $H_{min}$= negative sample minimum entropy,
                   $H_{max}$=negative sample maximum entropy,
                   $H_{-i}$ =negative sample entropy
5.   Output: subsets which each subset contains a number of negative samples
6.   Steps:
7.   FOR l = 1 to m
8.       $thrUp = H_{min} + \frac{l}{m}(H_{max} - H_{min})$
9.       $thrLow = H_{min} + \frac{l-1}{m}m(H_{max} - H_{min})$
10.       FOR i= 1 to N_
11.           IF thrLow ≤ $H_{-i}$ < thrUp
12.               the negative sample $x_i$ is distributed into the subset $sub_l$
13.      End FOR
14.  End FOR
15.  End
```

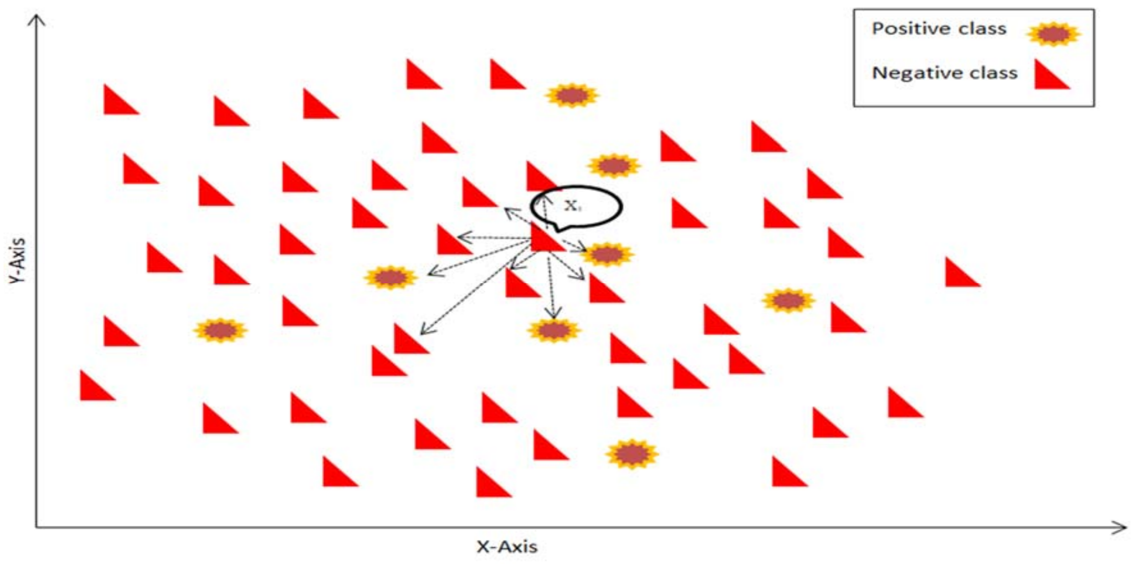

Fig. 4 Calculating Euclidean Distance of X_1

Next, the fuzzy memberships of samples in every subset are set as:

$$FM_l = 1.0 - (\beta * (l-1)), \quad l = 1,2 \ldots\ldots\ldots, m \qquad (10)$$

where FM_l is the fuzzy membership distributed in sub_l the fuzzy membership parameter $\beta \in (0, \frac{1}{(m-1)}]$. The fuzzy membership of instances FM_l in every subset is computed by Eq. (10) [9]. β should be greater than zero; if β =0, the fuzzy membership of the whole negative instances is equal to one. The whole negative instances contribute the same significance for training. Hence, $\beta \in (0, \frac{1}{(m-1)}]$.

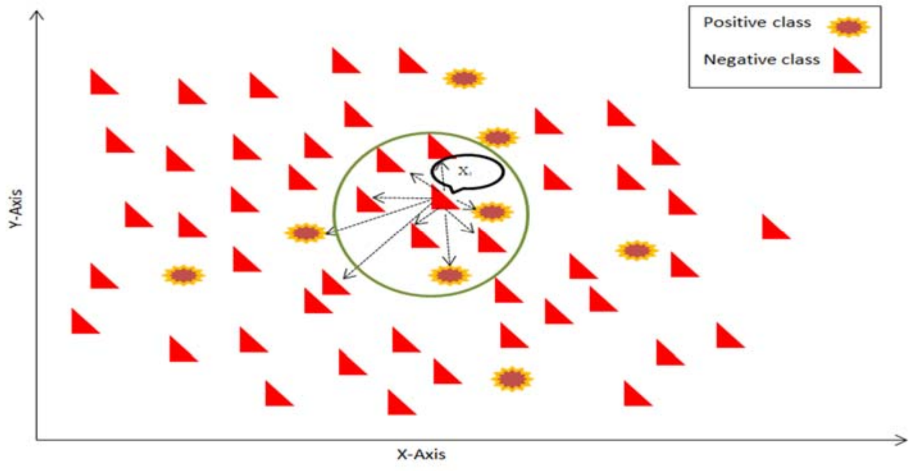

Fig. 5 Selecting k nearest neighbors of X_1

From the previous equation, we find that β controls in the FM_l range:

$$1 - (\beta * (m-1)) \le FM_l \le 1. \qquad (11)$$

It is discovered that a larger β value leads to a larger FM_l range. Finally, the fuzzy membership is assigned as [33]:

$$w_i = \begin{cases} 1.0 & \text{if } y_i = +1 \\ FM_l & \text{if } y_i = -1 \text{ and } x_i \in Sub_l \end{cases} \qquad (12)$$

where w_i is (weights) the fuzzy membership for each sample.

3.2. Entropy-Based Fuzzy Random Forest

Algorithm2 presents our proposed EFRF algorithm adopted from EFSVM algorithm [1].

3.2.1 Training process

Given the training set S $= \{ (x_i, y_i, w_i) \}_{i=1}^{N}$, where x_i is the n-dimensional sample, $y_i \in \{ +1, -1 \}$ refers to the predicted class label, and w_i is the entropy-based fuzzy membership(weights) that has been calculated by Eq. (12). Sample sets are created by random sampling from the training .Decision trees have been structured using the sample sets such as RF.

EFRF is used to access to the best decision that is able to divide the positive and negative class. Accessing to the best decision will be summarized in the above pseudocode.

Algorithm 2: The Proposed Algorithm (EFRF)

```
1.  Begin
2.  Input: The training sample S = { $x_i$, $y_i$ }$_{i=1}^{N}$, k = number of nearest neighbors k, m = number of
        subset m, and β = fuzzy membership parameter
3.      $y_i \in \{+1, -1\}$, $y_i = +1$: the sample $x_i$ belongs to the positive class, else belongs to the
        negative class,
4.  Output: Classifier
5.  Steps:
6.      Compute the k nearest neighbor for each negative sample $x_{-i}$.
7.      Enumerate the negative and positive samples in the k nearest neighbors of $x_{-1}$ $num_{+i}$ and
        $num_{-i}$ of $x_{-i}$.
8.      Compute the class probability of $x_{-i}$: $p_{+i} = \frac{num_{+i}}{k}$, and $p_{-i} = \frac{num_{-i}}{k}$
9.      Calculate the entropy for each negative sample, $x_{-i}$ , as: $H_i = -p_{+i} \ln(p_{+i}) - p_{-i} \ln(p_{-i})$
10.     Divide the negative sample into m subset according to algorithm 1.
11.     Assign (Weights) the fuzzy membership to each sample.
        $w_i = \begin{cases} 1.0, & \text{if } y_i = +1 \\ FM_l, & \text{if } y_i = -1 \text{ and } x_i \in Sub_l \end{cases}$
12. Function RandomForest(S,F)
13.         H ← ∅
14.         FOR i ∈ 1...B do
15.             S$^{(i)}$ ← A bootstrap Sample from S, W
16.             $h_i$ ← RandomizedTreeLearn (S$^{(i)}$, F)
17.             $H \leftarrow H \cup \{h_i\}$
18.         END FOR
19.         return H
20. End Function
```

```
21. Function RandomizedTreeLearn (S,F)
22.   At each node:
23.       f ← very small subset of F,   $S_T \leftarrow S(f)$
24.       Classifier ← $Grow_{DT\,(S_T, w_i)}$
25.       Test for leaf node;
26.       IF all cases in $S_T \in$ the same class, then return leaf with this class
27.       If Attributes is empty, then return leaf node with majority class
28.       Otherwise decision node:
29.       Search for the best decision attribute based on information gain
              $\Delta G = E(S_n) - \frac{|S_l|}{|S_n|} E(S_l) - \frac{|S_r|}{|S_n|} E(S_r)$
30.       Select this single attribute $a_i$ ,with outcomes { $o_1, o_2$…..}.
31.       Partition $S_T$ into $\{S_1, S_2\}$ according to outcomes.
32.       Apply recursively to: $Grow_{DT}(S_1), Grow_{DT}(S_2), \ldots$
33.       return classifier
34. End Function
35. Estimating class label: $\hat{y} = \arg Max_{Voting}(H, x_i)$
36. Compute error rate of classifier $\mathcal{E}_i$: $\varepsilon_t = \sum_{i:Yi \neq \widehat{y_i}}^{N} W_i^{(t)} / \sum_{i=1}^{N} W_i^{(t)}$
37. Compute Weight of classifier $\alpha_T$ : $\propto_t = \frac{1}{2} \log \frac{(M-1)(1-\varepsilon_t)}{\varepsilon_t}$
38. IF α > 0 then $W_i^{(t+1)} = \begin{cases} W_i^{(t)} \exp(\propto_t), & if \quad Yi \neq \hat{y_i} \\ W_i^{(t)} \exp(-\propto_t), & otherwise, \end{cases}$
        *Else* reject a classifier
39. End
```

3.2.1.1 Node splitting

Suggested technique training process is shown in Fig. 6 [19]. The slitting function chooses collections of features and thresholds in a random way that contain the best information gain. The information gain ΔG is figured by [9].

$$\Delta G = E(S_n) - \frac{|S_l|}{|S_n|}E(S_l) - \frac{|S_r|}{|S_n|}E(S_r) \qquad (13)$$

where S_n is set of instances at node n, S_l and S_r is sample set at left child node and sample set at right child node respectively, and E(S) is entropy calculated by [9].

$$E(S) = -\sum_{j=1}^{M} p(c_j)\log p(c_j) \qquad (14)$$

The samples are prioritized in calculating the information gain by highest weight and the class probability c_j, $p(c_j)$ which is figured using the sample weight i, W_i is presented using Eq. (15).

$$p(c_j) = \sum_{i\epsilon S \wedge Y_i = c_j} W_i \,/\, \sum_{i\epsilon S} W_i \qquad (15)$$

where the set of instances which reached to node has been assigned to S. A leaf node is generated when repetitive splitting develops the decision tree to a specific depth or till the information gain of a sample set that reaches a node is zero. The leaf node saves the probability of class P(c) calculated by Eq. (15) [19].

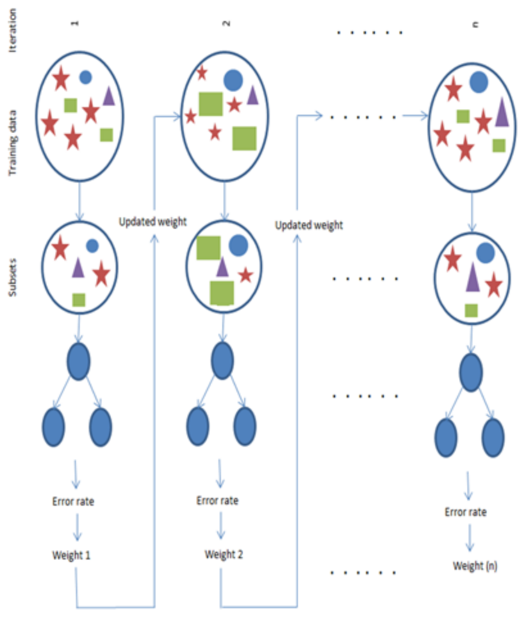

Fig. 6 Training Process Of The EFRF Method

3.2.1.2 Decision tree weighting

The decision tree weight, $\propto_t$, is computed by

$$\propto_t = \frac{1}{2}\log\frac{(M-1)(1-\varepsilon_t)}{\varepsilon_t} \qquad (16)$$

Where M is the class number and ε_t is the error rate of the decision tree. the error rate is calculated from the weights of the misclassified samples as[19]

$$\varepsilon_t = \sum_{i:Yi\neq\hat{y}_i}^{N} W_i^{(t)} \,/\, \sum_{i=1}^{N} W_i^{(t)} \qquad (17)$$

3.2.1.3 Updating training sample weights

Decision trees that facilely correct classification of the samples that are incorrectly classified in the next stage have been figured by making the weights of incorrectly classified samples large as [19]:

$$W_i^{(t+1)} = \begin{cases} W_i^{(t)}\exp(\propto_t), & if \quad Yi \neq \hat{y}_i \\ W_i^{(t)}\exp(-\propto_t), & otherwise, \end{cases} \qquad (18)$$

Where $\hat{y}_i$ is predicted class label using

$$\hat{y}_i = \arg\max p_t(c|l) \qquad (19)$$

After updating the weights of the training sample, the weights have been normalized to N. Updating the weights of the training sample is repeated until obtaining T weighted decision trees. After the whole decision trees are built, the weights have been normalized.

3.2.2Classification process

The samples which are used for testing are inputs to the whole of the decision trees, as displayed in Fig. 7 [19], and the probabilities of class which are arrived-at leaf nodes are output. Thereafter, the decision trees outputs,$p_t(c|\mathrm{x})$, are weight-averaged using the weights of decision tree as

$$P(c|\mathrm{x}) = \frac{1}{T}\sum_{t=1}^{T} \propto_t p_t(c|\mathrm{x}) \qquad (20)$$

The class which has the largest probability $\hat{y}_i$ is output as the classification result by[19]

$$\hat{y}_i = \arg\max p_t(c|x) \qquad (21)$$

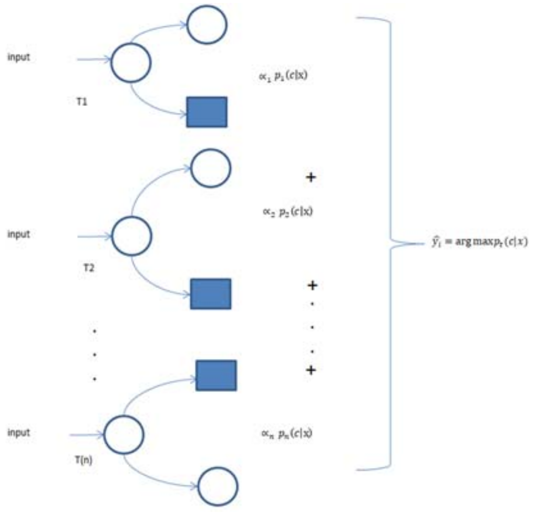

Fig. 7 Classification process of the EFRF method.

4. EXPERIMENTATIONS AND RESULTS

To analyze the accuracy of our solutions for the unbalanced dataset, three datasets are taken from UCI Machine Learning Repository [48] and KEEL repository[31]. The first one is CMC, which consists of 1473 examples and 9 features. The second is Haberman dataset, and it consists of 306 examples and 3 features. Glass5 is the last dataset used and it consists of 214 examples and 9 features. Table 2 describes the characteristics of each dataset, where min and maj refer to the instances belong to minor class and major class in data respectively, CL refers to the class label of minor class, and IR refers to the imbalanced ration.

Table2. UCI and KEEL Datasets Summary

Dataset	**A**	**S**	**Min/Maj**	**CL**	**IR**
Cmc	9	1473	333/1140	2	3.4234
Haberman	3	306	81/225	2	2.7778
Glass5	9	214	9/205	Positive	22.7777

We use a 5-fold stratified cross-validation partitioning scheme. For example: 5 random elements of instances with a 20% and a combination of 4 of them (80%) as a training set and the remaining as a test set. The results provided for each dataset have been the average results obtained by computing the mean of all partitions.

EFRF is implemented using python language based on the original RF. It is running on Intel Core(TM) i7-4800MQ CPU @2.70GHZ and 8.00G memory with Microsoft Windows 10. The experimentations parameters are as follows: max depth=11, subset=10, and numtrees=1000. The max depth, subset and num trees parameters present how the forest is structured. The num trees parameter defines how many trees compose the forest. Max depth defines the depth of each tree generated. The subset parameter indicates the number of subsets which the negative samples are separated into it.

Table 3 shows comparisons between our proposed EFRF algorithm and resampling techniques using precision, recall, $\beta - f -$ measure, GM, AUC, Cohen's kappa, balanced accuracy and MCC metrics. The bold font refers to the best result of each dataset. The results achieved by EFRF are stable according to the rest of the methods. From these results, we can clearly observe that for all the datasets, the proposed algorithm yielded the highest results on $\beta - f -$ measure, GM , recall , precision , Cohen's kappa , and balanced accuracy. AUC and MCC , for haberman, RF yielded the highest results, while for other datasets, cmc and glass 5, our proposed algorithm yielded the highest results.

The roc curves show approximate results for resampling approaches compared with our proposed algorithm which are illustrated in Figures.8 ,9, and 10.

Table3: Classification Results For UCI And KEEL Datasets Using Resampling Classifiers

		Original Rf	ROS+RF	RUS+RF	Smote+ RF	EFRF
Cmc	B—f—measure	0.35	0.538	0.573	0.470	**0.657**
	GM	0.545	0.661	0.658	0.625	**0.685**
	Recall	0.318	0.590	0.646	0.484	**0.661**
	Precision	0.583	0.397	0.351	0.421	**0.860**
	Cohen's kappa	0.300	0.283	0.238	0.276	**0.316**

	AUC	0.709	0.715	0.714	0.720	**0.737**
	Accuracy	0.626	0.666	0.658	0.645	**0.689**
	MCC	0.321	0.293	0.267	0.278	**0.325**
Haberman	B—f—measure	0.277	0.389	0.588	0.333	**0.778**
	GM	0.477	0.562	0.278	0.520	**0.718**
	Recall	0.25	0.375	0.295	0.312	**0.738**
	Precision	0.5	0.461	0.254	0.454	**0.915**
	Cohen’s kappa	0.192	0.233	0.020-	0.199	**0.332**
	AUC	**0.747**	0.752	0.513	0.732	0.703
	Accuracy	0.580	0.609	0.481	0.589	**0.718**
	MCC	**0.209**	0.235	0.053-	0.204	0.161
Glass5	B—f—measure	0.238	0.238	0.178	0.238	**1.0**
	GM	0.780	0.780	0.662	0.780	**1.0**
	Recall	**1.0**	**1.0**	0.452	**1.0**	**1.0**
	Precision	0.058	0.058	0.041	0.058	**1.0**
	Cohen’s kappa	0.069	0.069	0.035	0.069	**1.0**
	AUC	0.658	**1.0**	**1.0**	0.719	**1.0**
	Accuracy	0.804	0.804	0.719	0.804	**1.0**
	MCC	0.274	0.189	**0.135**	0.189	**1.0**

EFRF and cost-sensitive techniques performance are shown in Table 4 using precision, recall, $\beta - f -$ measure, GM, AUC, Cohen’s kappa, balanced accuracy, and MCC. The bold font points to the best result of each dataset. The results which are achieved by EFRF are stable according to the rest of the methods. Our approach achieves the best result on 2 datasets on $\beta - f -$ measure, on 3 datasets on GM, on 3 datasets on recall, on 3 datasets on precision, on 3 datasets on Cohen's kappa, on 3 datasets on AUC, on 3 datasets on balanced accuracy, and on 2 datasets on MCC. The roc curves show approximate results for cost-sensitive approaches compared with our proposed algorithm which are illustrated in Figures 11, 12, and 13.

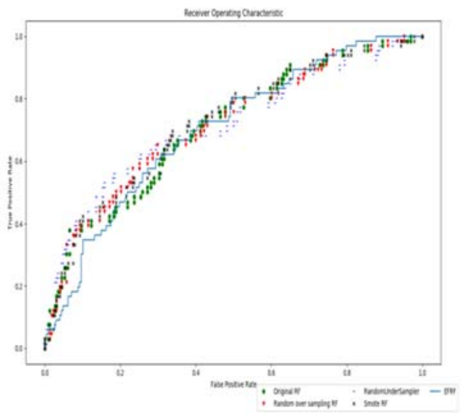

Fig.5. ROC For Cmc Dataset.

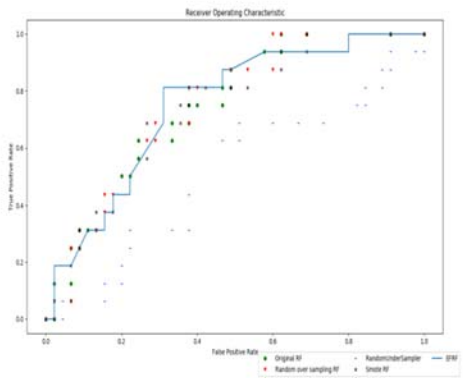

Fig.6. ROC For Haberman Dataset

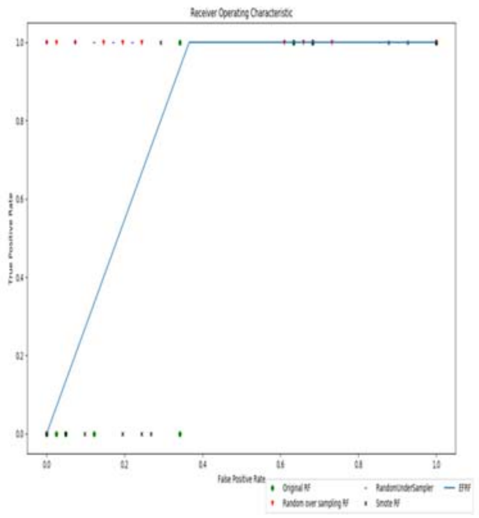

Fig.7. ROC For Glass5 Dataset

Table 4: Classification Results For UCI And KEEL Datasets Using Cost-Sensitive Classifiers

		Adaboost	GradientBoosting	RUSBoost	SmoteBoost	EFSVM	EFRF
Cmc	B—f—measure	0.220	0.386	0.371	0.339	0.652	**0.657**
	GM	0.425	0.567	0.554	0.527	0.0	**0.685**
	Recall	0.196	0.378	0.363	0.333	0.224	**0.661**
	Precision	0.419	0.423	0.406	0.366	0.706	**0.860**
	Cohen's kappa	0.145	0.238	0.218	0.172	0.0	**0.316**
	AUC	0.691	0.669	0.641	0.691	0.689	**0.737**
	Accuracy	0.559	0.614	0.605	0.583	0.593	**0.689**
	MCC	0.160	0.239	0.218	0.172	0.190	**0.325**
Haberman	B—f—measure	0.217	0.384	0.328	0.214	**0.783**	0.778
	GM	0.423	0.555	0.513	0.418	0.0	**0.718**
	Recall	0.187	0.375	0.312	0.187	0.262	**0.738**
	Precision	0.6	0.428	0.416	0.5	0.818	**0.915**
	Cohen's kappa	0.183	0.205	0.170	0.151	0.0	**0.332**
	AUC	0.416	0.686	0.576	0.402	0.679	**0.703**
	Accuracy	0.571	0.598	0.578	0.560	0.531	**0.718**
	MCC	**0.229**	0.206	0.173	0.178	0.063	0.161
Glass5	B—f—measure	0.238	0.25	0.208	0.227	0.317	**1.0**
	GM	0.780	0.796	0.732	0.765	0.0	**1.0**
	Recall	**1.0**	**1.0**	**1.0**	**1.0**	0.023	**1.0**
	Precision	0.058	0.062	0.05	0.055	0.409	**1.0**
	Cohen's kappa	0.069	0.076	0.052	0.062	0.0	**1.0**
	AUC	0.682	0.817	**1.0**	0.658	0.170	**1.0**
	Accuracy	0.804	0.817	0.768	0.792	**1.0**	**1.0**
	MCC	0.189	0.199	0.163	0.180	**1.0**	**1.0**

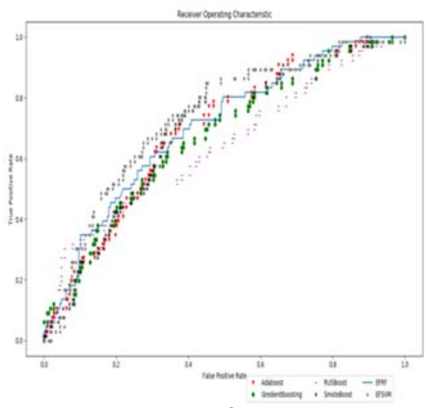

Fig.8. ROC for Cmc dataset

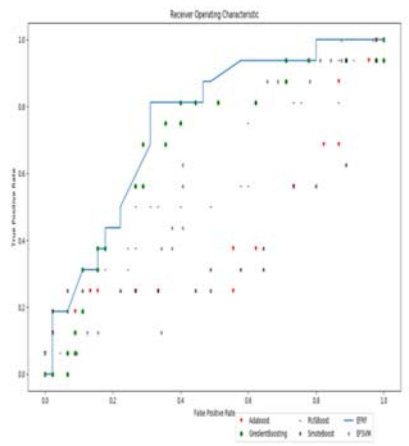

Fig.9. ROC for Haberman dataset

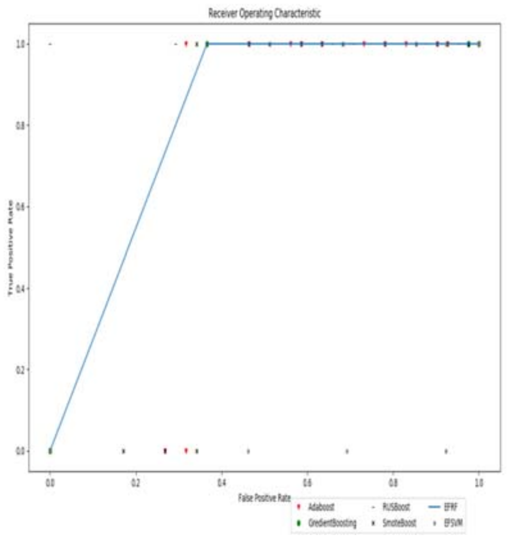

Fig.10. ROC for Glass5 dataset

5. CONCLUSION

Many standard classification techniques are leading to a weak performance on unbalanced trouble because they have been designed to deal only with balanced cases. Even though, RF has a strong performance on unbalanced datasets, it attaches the same significance to each training instance.

Computing the fuzzy membership to suits imbalanced classification problem is the key to robust classification algorithm that leverages the advantages of the ensemble of decision trees with the flexibility of fuzziness. To solve the inherent trouble of RF, many researchers applied fuzzy memberships to the training instances to show the various important of them. The base point is how to locate the fuzzy membership.

For addressing this challenge, we proposed EFRF algorithm that uses an EbFM valuation for unbalanced datasets. Virtually, it foremost computes the entropy of the negative samples in consideration of the corresponding nearest neighbors. Hence, the negative class is separated into various subsets based on the entropy. Eventually, assign a fuzzy membership (weights) to the samples in each subset.

For the positive class, a large fuzzy membership is assigned to each positive instance (for example: 1.0), to ensure the significance of the positive class. So, EFRF produces flexible decision surfaces than RF on the unbalanced datasets.

EFRF is tested on 3 UCI and KEEL datasets with using real diabetes mellitus clinical datasets. The experiments results on unbalanced datasets confirm that EFRF performs better than the compared algorithms. EFEF algorithm uses entropy to pay more attention to the samples with higher class certainty to result in more robust decision making to avoid losing information like other under sampling algorithms. It guarantees the importance of the positive samples through making the weight of all positive samples are equal to 1.0 which leads to iteration reduction.

The proposed algorithm is better than the compared algorithms which are depending on SMOTE. This is because SMOTE calculates knn based on positive samples only, while our proposed algorithm calculates it based on positive and negative samples together. Also, our proposed algorithm reduces the overfitting compared with other algorithms which depend on oversampling technique. Also, EFRF is better than EFSVM because EFRF uses RF which depends on bagging technique. Bagging idea is to create several subsets of data from training sample chosen randomly with replacement and each collection of subset data is used to train each decision trees. So, EFRF allows appearance of positive samples more than EFSVM.

Journal of Theoretical and Applied Information Technology
29th February 2020. Vol.98. No 04
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Overall, the proposed algorithm showed promising results compared to other imbalanced classification techniques including EFSVM technique [9] that we adopted our proposed algorithm from it. It featured both high precision and high recall which makes it a perfect choice for security-wise application.

6. FUTURE WORK

In the future , we will use the map reduce technique for "classification the imbalanced Big Data" using our proposed classifier.

REFRENCES:

[1] Mena, L.J. and J.A. Gonzalez. *Machine Learning for Imbalanced Datasets: Application in Medical Diagnostic*. in *Flairs Conference*. 2006.

[2] Bendovschi, A., *Cyber-attacks–trends, patterns and security countermeasures.* Procedia Economics and Finance, 2015. **28**: p. 24-31.

[3] Huang, C., et al., *Deep imbalanced learning for face recognition and attribute prediction.* IEEE transactions on pattern analysis and machine intelligence, 2019.

[4] Perera, B.K., *A class imbalance learning approach to fraud detection in online advertising.* Masdar Institute of Science and Technology, 2013.

[5] Breiman, L., *Random forests.* Machine learning, 2001. **45**(1): p. 5-32.

[6] Freund, Y., R. Schapire, and N. Abe, *A short introduction to boosting.* Journal-Japanese Society For Artificial Intelligence, 1999. **14**(771-780): p. 1612.

[7] Freund, Y. and R.E. Schapire. *Schapire R: Experiments with a new boosting algorithm*. in *In: Thirteenth International Conference on ML*. 1996. Citeseer.

[8] Guelman, L., *Gradient boosting trees for auto insurance loss cost modeling and prediction.* Expert Systems with Applications, 2012. **39**(3): p. 3659-3667.

[9] Fan, Q., et al., *Entropy-based fuzzy support vector machine for imbalanced datasets.* Knowledge-Based Systems, 2017. **115**: p. 87-99.

[10] Chawla, N.V., N. Japkowicz, and A. Kotcz, *Special issue on learning from imbalanced data sets.* ACM Sigkdd Explorations Newsletter, 2004. **6**(1): p. 1-6.

[11] Batista, G.E., 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, 2004. **6**(1): p. 20-29.

[12] Chawla, N.V., et al., *SMOTE: synthetic minority over-sampling technique.* Journal of artificial intelligence research, 2002. **16**: p. 321-357.

[13] Garcı, S., et al., *Evolutionary-based selection of generalized instances for imbalanced classification.* Knowledge-Based Systems, 2012. **25**(1): p. 3-12.

[14] Zadrozny, B., J. Langford, and N. Abe. *Cost-Sensitive Learning by Cost-Proportionate Example Weighting*. in *ICDM*. 2003.

[15] Galar, M., et al., *EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling.* Pattern Recognition, 2013. **46**(12): p. 3460-3471.

[16] Maldonado, S. and J. López, *Imbalanced data classification using second-order cone programming support vector machines.* Pattern Recognition, 2014. **47**(5): p. 2070-2079.

[17] Bunkhumpornpat, C., K. Sinapiromsaran, and C. Lursinsap. *Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem*. in *Pacific-Asia conference on knowledge discovery and data mining*. 2009. Springer.

[18] Zhao, P., et al. *Cost-sensitive online classification with adaptive regularization and its applications*. in *2015 IEEE International Conference on Data Mining*. 2015. IEEE.

[19] Mishina, Y., et al., *Boosted random forest.* IEICE Transactions on Information and systems, 2015. **98**(9): p. 1630-1636.

[20] Verikas, A., A. Gelzinis, and M. Bacauskiene, *Mining data with random forests: A survey and results of new tests.* Pattern recognition, 2011. **44**(2): p. 330-349.

[21] Breiman, L., *Bagging predictors.* Machine learning, 1996. **24**(2): p. 123-140.

[22] Breiman, L., *Using adaptive bagging to debias regressions.* 1999, Technical Report 547, Statistics Dept. UCB.

[23] Barandela, R., et al., *Strategies for learning in class imbalance problems.* Pattern Recognition, 2003. **36**(3): p. 849-851.

[24] Kubat, M. and S. Matwin. *Addressing the curse of imbalanced training sets: one-sided selection*. in *Icml*. 1997. Nashville, USA.

[25] He, H. and E.A. Garcia, *Learning from imbalanced data.* IEEE Transactions on knowledge and data engineering, 2009. **21**(9): p. 1263-1284.

[26] Batuwita, R. and V. Palade, *Adjusted geometric-mean: a novel performance measure for imbalanced bioinformatics datasets learning.* Journal of Bioinformatics and Computational Biology, 2012. **10**(04): p. 1250003.

[27] Viera, A.J. and J.M. Garrett, *Understanding interobserver agreement: the kappa statistic.* Fam med, 2005. **37**(5): p. 360-363.

[28] Hoens, T.R. and N.V. Chawla, *Imbalanced datasets: from sampling to classifiers.* Imbalanced Learning: Foundations, Algorithms, and Applications, 2013: p. 43-59.

[29] Vintr, T., et al. *Spatiotemporal models of human activity for robotic patrolling*. in *International Conference on Modelling and Simulation for Autonomous Systesm*. 2018. Springer.

[30] Weng, C.G. and J. Poon. *A new evaluation measure for imbalanced datasets*. in *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*. 2008. Australian Computer Society, Inc.

[31] Alcalá-Fdez, J., et al., *Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework.* Journal of Multiple-Valued Logic & Soft Computing, 2011. **17**.