Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 5, No. 3, August 2020, Pp. 203-210

The comparison of imbalanced data handling method in software defect prediction

Khadijah*[1], Priyo Sidik Sasongko[2]
Department of Informatics, Universitas Diponegoro, Indonesia[1,2]

Article Info
Keywords:
Extreme Learning Machine (ELM), Software Defect Prediction, SMOTE, Weighted-ELM

Article history:
Received 11 February 2020
Revised 05 May 2020
Accepted 13 August 2020
Published 31 August 2020

Cite:
Khadijah, K., & Sasongko, P. (2020). Software Defect Prediction Using Synthetic Minority Over-sampling Technique and Extreme Learning Machine. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, 5(3). doi:https://doi.org/10.22219/kinetik.v5i3.1049

*Corresponding author.
khadijah
E-mail address:
khadijah@live.undip.ac.id

Abstract
Software testing is a crucial process in software development life cycle which will affect the software quality. However, testing is a tedious task and resource consuming. Software testing can be conducted more efficiently by focusing this activitiy to software modules which is prone to defect. Therefore, an automated software defect prediction is needed. This research implemented Extreme Learning Machine (ELM) as classification algorithm because of its simplicity in training process and good generalization performance. Aside classification algorithm, the most important problem need to be addressed is imbalanced data between samples of positive class (prone to defect) and negative class. Such imbalance problem could bias the performance of classifier. Therefore, this research compared some approaches to handle imbalance problem between SMOTE (resampling method) and weighted-ELM (algorithm-level method).The results of experiment using 10-fold cross validation on NASA MDP dataset show that including imbalance problem handling in building software defect prediction model is able to increase the specificity and g-mean of model. When the value of imbalance ratio is not very small, the SMOTE is better than weighted-ELM. Otherwise, weighted-ELM is better than SMOTE in term of sensitivity and g-mean, but worse in term of specificity and accuracy.

1. Introduction

Software testing is a crucial process in software development life cycle which is aimed to find errors of software before it is deployed to end user [1]. The results of software testing will affect the software quality. Incomplete software testing can result in bad software quality. However, comprehensive software testing activities need big resources in term of cost, human and time, especially if the developed software is complex and large scale [2]. Software testing can be conducted more efficiently by focusing this activitiy to software modules which is prone to defect. Then, the testing process can be more focused to those parts. One of the approaches for predicting software defect is based on software metrics that describe the properties of software [3]. Therefore, an automated software defect prediction model can be built based on those software metrics. In term of machine learning, such kind of model can be built by using supervised learning or classification algorithm.

The various classification algorithm have been implemented for predicting software defect, such as Artificial Neural Network (ANN) [4][5], Support Vector Machine (SVM) [6] and Naïve-Bayes classifier [7]. Naïve-Bayes classifier assumes that every attribute (software metric) is conditional independence to the class of software defect [8], while in the case of software defect prediction these attributes is not truly conditional independence to the class of software defect. When using ANN, gradient-descent learning algorithm is usually implemented. This learning algorithm usually needs many iteration to reach convergency and involves many parameters in its training process. The other learning algorithm for ANN that can solve the limitation of gradient-descent learning is extreme learning machine (ELM). The training process of ELM is extremely fast because it only needs one iteration. ELM also involves less number of parameter in its training process [9]. ELM is also faster and more scalable than SVM. Moreover in binary classification case the generalization performance of ELM is better than ANN and similar to SVM [10].

In spite of classification algorithm, the other important problem in software defect prediction is the imbalance ratio between number of samples inpossitive class (prone to defect) and negative class. The value of imbalance ratio can reach 1/100, even 1/1000 in certain dataset. Classifications on imbalanced dataset can bias the performance of the resulting classifier. The classifier usually is able to achievehigh accuracy, but not in sensitivity. For example, when number of samples of positive class is 10 and number of negative class is 90 and classifier always classifies each sample into negative class, then the classifier can achieve 90% accuracy. However, it can not be claimed that it is a good classifier because it never classifies a sample into possitive class, as the positive class is the topic of interest. Therefore, it is needed an additional method for handling imbalanced data distribution in order to get optimal results [11].

There are three strategies for handling the problem of imbalanced data: i) by modifying the dataset (data-lavel or resampling method); ii) by modifying classification algorithm (algorithm-level method); iii) by modifying the misclassification cost (cost sensitive method). The third strategy is used less frequently that the fisrt and second strategy because the difficulties in estimating the value of misclassification cost correctly. Resampling method modifies the distribution of dataset to achieve balance ratio between number of positive class and negative class. The advantages of this method is its flexibility because it is independence to the classification algorithm [12]. Synthetic Minority Over-sampling Technique (SMOTE) is one of the popular resampling methods that has been frequently used in many cases [13]. Some modification of SMOTE have also been developed, but experiments show that the performance of original or modified version of SMOTE is subjected to the dataset and classification algorithm. SMOTE is able to reach better performance than the modified version in certain dataset and classification algorithm [14].Therefore, this research uses SMOTE which is the benchmark of oversampling method that have been successfully applied in many cases to improve the generalization performance of classifier in minority class [14][15][16]. SMOTE even still has good performance when the number of samples in minority class is quite small [11].

In contrast to resampling method,algorithm-level method does not modify the dataset, but modifies the classification algorithm directly [12]. Since this research implements ELM classification algorithm, the modification of ELM which is designed to cope directly with imbalanced data problem, namely weighted-ELM, is implemented in this research. Weighted-ELM introduced weighted matrix in the calculation of output weights that is aimed to move boundary line toward to the area of majority class in order to increase the generalization performance in the minority class.Experiment results show that weighted-ELM has better performance than original ELM as classifier on imbalanced dataset [17].

This research is aimed to create and compare models for predicting software defect by using three scenarios: i) directly classification using ELM (without imbalanced data handling); ii) resampling dataset using SMOTE followed by classification using ELM (SMOTE-ELM); iii) directly classification using weighted-ELM that have encountered imbalanced data handling in its algorithm. The resulting models are then evaluated to measure their performance.

2. Research Method

There are two process involved to build a classification model for software defect prediction, the training and testing process. The training is aimed to create a model, while testing is aimed to evaluate the resulting model and select the best model. Figure 1 shows the flowchart for the second scenario (SMOTE-ELM). In training process, over-sampling is applied before classification to balance the dataset. When applying the first scenario (ELM without resampling) and the third scenario (weighted ELM) the process of oversampling is removed, while in the third scenario ELM is replaced by weighted-ELM.

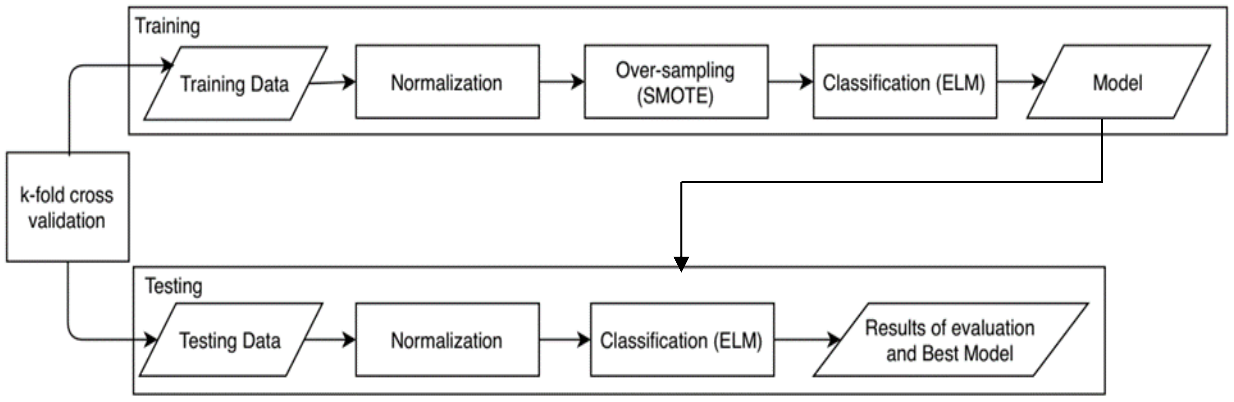


Figure 1.The Research Flowchart

2.1 Dataset

This research used dataset from NASA MDP dataset PC1, PC2, PC3 and PC4 in cleaned data version [18]. Those dataset can be downloaded in https://github.com/klainfo/NASADefectDataset.The description of each dataset are shown at Table 1 where d is the number of feature attributes and N is number of samples. The imbalance ratio is ratio between number of samples in positive class (prone to defect) and number of samples in negative class. Each dataset consists of some feature attributes and one decision attribute. The feature attributes are numeric software metrics that are used as an input feature for prediction, while the decision attribute is the output class which shows the existence of software defect (yes or no).

Table 1. Description of Dataset

Dataset	d	N	Number of negative samples	Number of positive samples	Imbalance ratio
PC1	37	735	674	61	0.0905
PC2	36	1493	1477	16	0.0108
PC3	37	1099	961	138	0.1436
PC4	37	1380	1201	179	0.1490

2.2 Normalization

Normalization is aimed to scale the value of each feature in the dataset into a specific range and prevent overweighing from feature with large range value to feature with small range value. Each feature in the dataset are normalized into [-1,1] using min-max normalization as suggested by [10].For target or decision attribute, the value is set to 1 if it belongs to positive class (prone to defect) and -1 if it belongs to negative class.

```
Function SMOTE(Sample[1..T][1..numattrs] :arrayofarrayofreal, T : integer, numattrs: integer, N : integer, k: integer)
→<arrayofarrayofreal>
{This function is used for over-sampling minority class samples. The parameters of this function are:
Sample[][] is array original minority class samples
T is number of samples in minority class
numatrrs is the number of sample's attributes
N is percentage of over-sampling (N ≥ 100%)
k is number of nearest neighbors
Local Dictionary:
  i : integer {counter for original sample}
  nnarray : arrayofinteger {array for storing index of sample's nearest neighbors}
  nn: integer {index of the choosen nearest neighbor of a sample}
  attr : integer {counter for sample's attributes}
  dif : real {the difference of attribute value between synthetic sample and original samples}
  gap : real {random value between 0..1 for multiplying dif}
  newindex : integer {counter for generated synthetic samples}
  Synthetic : arrayofarrayofreal {array of generated synthetic samples}
Algorithm:
  {initialization}
  newindex← 0

  i traversal [1..T]
        {obtain k nearest neighbor for i-th sample using Euclidean distance formula and store location index of its
        nearest negihbors into nnarrays}
        <nnarray>←compute_nearest_neighbor(Sample[i][1..numattrs])
        {generate synthetic samples}
        N ← (integer)(N/100)

        while N ≠ 0
            {choose a sample from k nearest neighbors of i-th sample randomly }
            nn← random(1,k) {nn is random value between  1..k}

            attr traversal [1..numattrs]
                {calculate the difference between attribute value of sample's nearest neighbor and its sample}
                dif← Sample[nnarray[nn]][attr] – Sample[i][attr]
                gap ← random(0,1) {gap is random value 0..1}
                Synthetic[newindex][attr] = Sample[i][attr] + gap*dif
            {end traversal attr}

            newindex←newindex + 1
            N ← N-1
        {endwhile N ≠ 0}

  {end traversal i}

  →Synthetic[0..newindex-1][1..numattrs]
```

Figure 2. Pseudocode of SMOTE [13]

2.3 Synthetic Minority Over-sampling Technique (SMOTE)

SMOTE is applied to balance the ratio of samples in posstive class and negative class. SMOTE creates a synthetic sample from minority class by choosing a point randomly in the line segment connecting a sample and its nearest neighbor. Figure 2 shows the pseudocode of SMOTE algorithm. There are two parameters in this algorithm. The first parameter is N, the amount of over-sampling. The second parameter is k, the number of nearest neighbor selected for creating synthetic samples. For example, if the number of samples in minority class is 10, the value of N is set to 200% and the value of k is set to 5, SMOTE will creates 2 synthetic samples for every samples in the minority class. Therefore, the total number of synthetic sample is 2 x 10 = 20 and the total number of samples in minority class now is 10 + 20 = 30 [13].

2.4 Extreme Learning Machine (ELM)

ELM is a learning algorithm for artificial neural network which has single hidden layer and feedforward architecture. The learning algorithm of ELM only needs one iteration, so that the training process is extremely fast [9][10]. Figure 3 shows the architecture of artificial neural network used in this research. This network has d input nodes thatare equal to number of feature attributes in each dataset, L hidden nodesdanone output node that represents the decision attribute or existence of defect. This network apply radial basis function (RBF) hidden node, so thatparameter $(\mathbf{a}_j, b_j)_{j=1}^{L}$from input nodesto hidden nodesarecenter vector$\mathbf{a}_j$andscaling parameter$b_j$in the j-th hidden node. The weights from hidden nodesto output node is vector$\boldsymbol{\beta} = [\beta_1, \dots, \beta_L]^T$[19].

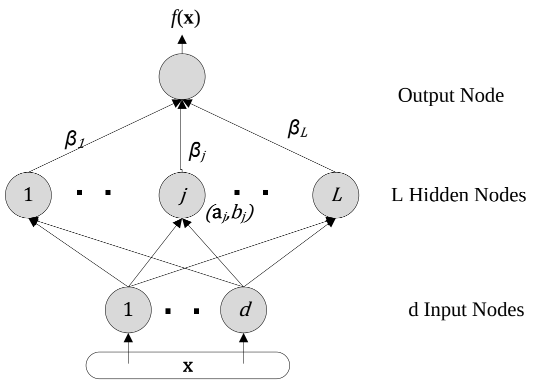


Figure 3. The Architecture of Network for ELM [19]

Training algorithm of ELM uses Npairs of training sample, $(\mathbf{x}_i, t_i)_{i=1}^{N}$where$\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{id}]^T \in R^d$representfeature vector and$t_i$represent target attribute of i-th sample. The training algorithm of ELM is described as follows [10]:

[1] Initialize parameter $(\mathbf{a}_j, b_j)_{j=1}^{L}$fromeach input node to$1..L$ hidden nodes randomly.

[2] Calculate the matrix$\mathbf{H}$ (the output of eachhidden node$1..L$for all training samples from$1..N$)using a specific activation function as Equation 1. This research used multiquadric activation function as shown in Equation 2.

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_1) & \dots & h_L(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ h_1(\mathbf{x}_N) & \dots & h_L(\mathbf{x}_N) \end{bmatrix}_{N \times L} \quad (1)$$

$$h_j(\mathbf{x}_i) = (\|\mathbf{x} - \mathbf{a}_j\|^2 + {b_j}^2)^{1/2} \quad (2)$$

[3] Calculate vector $\boldsymbol{\beta} = [\beta_1, \dots, \beta_L]^T$, the weights from hidden nodesto output node, as Equation 3. The vector$\mathbf{T}$is output vector containing target value for each training samples from $1..N$, $\mathbf{I}$isidentity matrix, and$C$isregularization parameter to increase the generalization performance.

$$\boldsymbol{\beta} = \begin{Bmatrix} \mathbf{H}^{\mathrm{T}} \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^{\mathrm{T}} \right)^{-1} \mathbf{T} & \text{if } N < L \\ \left(\frac{\mathbf{I}}{C} + \mathbf{H}^{\mathrm{T}}\mathbf{H} \right)^{-1} \mathbf{H}^{\mathrm{T}}\mathbf{T} & \text{if } N \gg L \end{Bmatrix} \quad (3)$$

In the testing process, class of an input sample $\mathbf{x}$ can be predicted by using Equation 4 followed by Equation 5. Equation 4 is aimed to calculate the value of output network using parameter $(\mathbf{a}_j, b_j)_{j=1}^{L}$ and $\boldsymbol{\beta}$ and also activation function from the training process, while Equation 5 is aimed to predict the class of output $f(\mathbf{x})$ [10].

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} \tag{4}$$

$$class(\mathbf{x}) = \mathrm{sign}(f(\mathbf{x})) \tag{5}$$

ELM algorithm is not especially designed for used in imbalanced dataset. Therefore, in the imbalanced dataset to maximize the accuracy, the boundary line of classifier usually is movedtoward to the side of minority class. As a result, the generalization performance of classifier in the majority class is high, while in the minority class is low. But, the low performance in the minority class does not give significant effect on the end accuracy of classifier because the number of samples in the minority class is small enough. Therefore, ELM is modified into weighted-ELM which is especially designed for handling imbalanced data distribution [17].

Weighted-ELM modifies the training algorithm of ELM by adding diagonal weight matrix$\mathbf{W} = diag\{w_{ii}\}$, $i = 1, \dots, N$. Each row of matrix $\mathbf{W}$ is associated to each sample $\mathbf{x}_i$ in the training data. The addition of this matrix is aimed to move boundary line toward to the area of majority class in order to increase the generalization performance in the minority class. But how much the boundary line is moved, should be carefully decided so that the generalization performance in the majority class does not decrease. Therefore, the value of w_{ii} is calculated using the golden ratio, 0,618:1 as shown at Equation 6 for binary class problem. Then, the output weight matrix can be calculated by using Equation 7 [17].

$$w_{ii} = \begin{cases} \dfrac{0.618}{number\ of\ samples\ in\ majority\ class} & , if\ sample\ \mathbf{x}_i \text{is in majority class} \\ \dfrac{1}{number\ of\ samples\ in\ minority\ class} & , if\ sample\ \mathbf{x}_i \text{is in minority class} \end{cases} \tag{6}$$

$$\boldsymbol{\beta} = \begin{Bmatrix} \mathbf{H}^{\mathrm{T}}\left(\dfrac{\mathbf{I}}{C} + \mathbf{WHH}^{\mathrm{T}}\right)^{-1}\mathbf{WT} & \text{if } N < L \\ \left(\dfrac{\mathbf{I}}{C} + \mathbf{H}^{\mathrm{T}}\mathbf{WH}\right)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{WT} & \text{if } N \gg L \end{Bmatrix} \tag{7}$$

2.5 Evaluation Metrics

Binary classifier can be evaluated based on its accuracy, sensitivity and specificity. Accuracy is the comparison between number of samples predicted correctly by classifer and the total number of samples. Sensitivity is the accuracy of positive class, while specificity is the accuracy of negative class [8].The ratio between number of samples in positive class and negative class in the NASA MDP dataset is imbalanced, so that the use of accuracy as an evaluation metric is not quite representative. Therefore, in this research classifier is also evaluated by $g - mean$ as shown at Equation 8. The value of $g - mean$ is not influenced by the balanceness of dataset because it considers the performance of classifier in both class, positive and negative [20].

$$G - mean = \sqrt{sensitivity \times specifity} \tag{8}$$

3. Results and Discussion

In order to get comparating result, in this research the classifier for predicting software defect was built by using three scenarios: 1) directly classification using ELM; 2) over-sampling minority class using SMOTE followed by classification using ELM; 3) directly classification using modification of ELM, weighted-ELM. This experiment applied 10-cross validation to divide dataset into training and testing data. First, dataset is divided into 10 parts where each part has the same ratio of positive samples and negative samples, then each experiment uses 9 part as training data and 1 part as testing data. The experiment is repeated until 10 times using different part as testing data, then the reported result is the mean of result of 10 experiments [21].

There are two parameters in the ELM training algorithm. They are number of hidden nodes L and regularization parameter C. In this research, number of hidden nodes is set to 1000. Some prior research used 1000 hidden nodes for various dataset and concluded that as long as the number of hidden nodes is quite large, it does not make significant effect to the end result [10]. Therefore, there is only one remaining parameter to be adjusted, regularization parameter C. The value of$C$ is search from range $\{2^1, 2^2, \dots, 2^{24}, 2^{25}\}$ to obtain optimal result.

In the over-sampling using SMOTE, there are also two parameters, they are number of nearest neighbors$k$and percentage of over-sampling N. The value of N in PC1, PC2 and PC3 dataset is set into certain number to make the ratio of positive and negative samples in training data close to 1. Then, the value of k is set to 10 because in each dataset there are 6 to10 synthetic samples that would be created for each minority sample. In PC2 dataset the value k is set 12 because there are only 12 to 13 samples of positive or minority class in the training data and the maximum percentage of over-sampling is equal to number of nearest neighbor parameter since it only creates a synthetic sample for each nearest negihbor. As a result, the number of oversampled samples from minority class is still lower than the number of samples from majority class, so that the oversampled PC2 dataset are remain imbalanced. Table 2 shows the combination of SMOTE parameters in four datasets.

Table 2. The Combination Value of SMOTE Parameters

dataset	k	N
PC1	10	1000
PC2	12	1200
PC3	10	600
PC4	10	500

The results of experiment using ELM, SMOTE-ELM and weighted ELM for all dataset are shown at Table 3, Table 4, and Table 5, respectively. Then, the graphic at Figure 4 show the comparison of g-mean and accuracy.

Table 3. The Results of Experiment Using ELM

Dataset	C	Accuracy	Sensitivity	Specificity	G-mean
PC1	25	85.83% ± 3.43%	42.62% ± 13.86%	89.75% ± 2.89%	61.10% ± 11.26%
PC2	25	97.19% ± 1.18%	30.00% ± 42.16%	97.97% ± 1.01%	33.82% ± 44.71%
PC3	24	83.53% ± 2.43%	36.21% ± 12.65%	90.33% ± 3.02%	56.25% ± 10.18%
PC4	20	86.80% ± 2.61%	42.75% ± 10.76%	93.34% ± 3.04%	62.60% ± 8.30%

Table 4. The Results of Experiment Using SMOTE-ELM

Dataset	C	Accuracy	Sensitivity	Specificity	G-mean
PC1	7	82.19% ± 6.87%	72.38% ± 24.31%	83.07% ± 7.11%	76.40% ± 14.00%
PC2	9	98.06% ± 1.16%	40.00% ± 45.95%	98.78% ± 0.95%	43.96% ± 47.54%
PC3	8	77.52% ± 4.41%	73.90% ± 13.13%	78.04% ± 4.23%	75.68% ± 7.67%
PC4	9	83.97% ± 4.16%	82.68% ± 12.41%	84.19% ± 5.19%	83.11% ± 5.79%

Table 5. The Results of Experiment Using Weighted-ELM

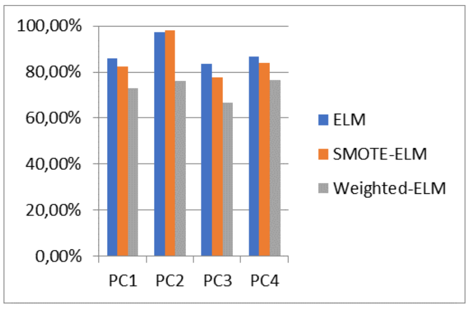
Dataset	C	Accuracy	Sensitivity	Specificity	G-mean
PC1	12	73.05% ± 6.64%	86.67% ± 13.15%	71.78% ± 7.57%	78.47% ± 6.23%
PC2	4	76.09% ± 3.55%	85.00% ± 24.15%	76.04% ± 3.66%	79.42% ± 11.89%
PC3	12	66.69% ± 4.91%	84.01% ± 11.32%	64.20% ± 4.69%	73.32% ± 6.63%
PC4	16	76.65% ± 3.58%	94.41% ± 5.86%	74.03% ± 4.57%	83.48% ± 2.22%

The results on Table 3 show that directly classification using ELM without any handling for imbalanced data problem in all dataset, tend to achieve the highest accuracy and the lowest g-mean, compared to the other methods (SMOTE-ELM and weighted-ELM)as also shown by Figure 4. It is caused by the high value of specificity, but the low value of sensitivity, so that the the value of g-mean becomes low. However, because the number of positive samples is far smaller than the number of negative samples, especially in PC2 dataset, the value of accuracy remains high.

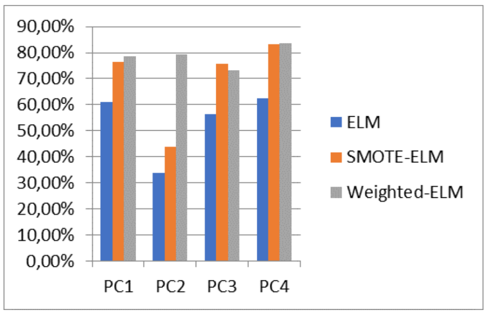
In the second scenario, when SMOTE is added for over-sampling dataset before classification, it can be observed that the value of g-mean and sensitivity in all dataset are higher compared to the first scenario, but the value of accuracy and specificity decrease, except on PC2 dataset. In this case, the value of sensitivity is still lower than the value of specificity. This condition show that SMOTE is able to increase the generalization performance on positive class, but it is also followed by slightly drecreasing generalization performance on negative class. It can be observed that the increasement values of g-mean are higher than the decreasement values of accuracy from ELM to SMOTE-ELM. Otherwise, in PC2 dataset, the value of sensitivity and g-mean is still below 50% because the imbalance ratio in PC2 dataset is very small. This can be happen because SMOTE creates synthetic samples by interpolating from the nearest neighbors of the selected samples in minority class. As a result, when the number of minority class is very low, there are only few synthetic samples can be generated because of the limited number of nearest neighbor samples as a based to create synthetic samples, so that the number of oversampled minority class samples are still to far from the number of majority class samples. On the other word, the oversampled dataset remain imbalanced.

Same as the second scenario, when using weighted-ELM the generalization performance on positive class (sensitivity) also increase, but it is also followed by drecreasing generalization performance on negative class (specificity). The value of sensitivity is even higher than the value of specificity (contrast to the result of the first and second scenario).Compared to SMOTE-ELM, theincreasement of sensitivity from the fisrt scenario is higher, but the thedecreasement of specificity is also higher. The values of g-mean are slightly lower on PC3 dataset, slightly higher on PC1 and PC4 dataset, and higher on PC2 dataset. Otherwise, the accuracy of weighted-ELM are lower than accuracy of SMOTE-ELM in all dataset. On PC2 dataset, the value of imbalance ratio is very small so that in the learning algorithm of weighted-ELM the minority samples are given high value in the calculation of weight matrix $\mathbf{W}$. As a result, the boundary line is moved toward to the area of majority class. Then, it causes the generalization performance of minority class increase, but the generalization of majority class decrease. Because the number of majority class samples are far higher than the number of minority class samples, the value of accuracy also decreases.

From this comparison, it can be concluded that in this software prediction case, incuding the imbalanced data handling is able to improve the performance of classifier in term of sensitivity and g-mean, but is also followed by slightly decreasing performance in term of specificity and accuracy. SMOTE-ELM is better than weighted-ELM, especially when the imbalance ratio is not very small (in PC3 and PC4 dataset). Otherwise, when the imbalance ratio is very small, especially in PC2 dataset, SMOTE is not able to reach good sensitivity and g-mean. When the value of imbalance ratio is very small, weighted-ELM can achived highersensitivity and g-mean compared to SMOTE-ELM, but lower specificity and accuracy. Generally, weighted-ELM is able to much increase the generalization performance in minority class, but as a consequence it also much decrease the generalization performance in majority class. While, when using SMOTE the incresement of generalization performance in minority class and the decrasement of generalization performance in majority class are smoother than in weighted-ELM.



(a)



(b)

Figure 4. The Comparison of g-mean (a) and Accuracy (b) in All Dataset

The comparison with other research can be seen in Table 6. Generally, the imbalance handling method used in this research (both, SMOTE and weighted-ELM) is able to reach better g-mean than in the other research [5][22] (except for PC2 dataset in SMOTE-ELM), but most of the accuracy achieved by this research is still lower. This condition can happen because the increasing permormance of classifier in minority class are still followed by the decrasing performance in majority class.

Table 6. The Comparison with Other Researches

Metric	Dataset	SMOTE-ELM	Weighted-ELM	Logistic Regression and Resampling [22]	Neural Network and Random Undersampling [5]
G-Mean	PC1	76.40%	78.47%	55.10%	65.00%
	PC2	43.96%	79.42%	-	73.00%
	PC3	75.68%	73.32%	43.80%	66.00%
	PC4	83.11%	83.48%	73.50%	62.00%
Accuracy	PC1	82.19%	73.05%	93.41%	87.80%
	PC2	98.06%	76.09%	-	69.40%
	PC3	77.52%	66.69%	86.31%	61.00%
	PC4	83.97%	76.65%	90.64%	84.60%

4. Conclusion

In this research the classifiers for software defect prediction was built. For building the classifier model, four dataset from NASA MDP were used, they are PC1, PC2, PC3 and PC4 dataset. One of the important problems in those dataset is the imbalance data distribution between samples of positive class and samples of negative class. Therefore, in this research a method for handling this imbalance data problem was added. In order to get comparating result, the classifierswere built by using three scenarios: 1) directly classification using ELM; 2) over-sampling minority class using SMOTE followed by classification using ELM; 3) directly classification using modification of ELM, weighted-ELM.

The results of experiment using 10-fold cross validation show that directly classification using ELM obtain the worse result compared to the other scenarios. Therefore in this software defect prediction case, it can be concluded that incuding the imbalanced data handling is able to improve the generalization performance of classifier in minority class. When the value of imbalance ratio is not very small the SMOTE-ELM is better than weighted-ELM because SMOTE-ELM can achieve higher value, both in g-mean and accuracy. When the value of imbalance ratio is very small, weighted-ELM is better than SMOTE-ELM in term of sensitivity and g-mean, but worse than SMOTE-ELM in term of specificity and accuracy.

The future research can improve the way to handle imbalance problems so as not to worsen performance in the majority class. In addition, the future research can also examine the method for selecting features or attributes that play signigicant role to the prediction class. The reduction of number of features as input to a classfieris expected to increase the performance of classifier.

References

[1] R. S. Pressman, *y Software Engineering: A Practitioner's Approach Seventh Edition*. McGraw-Hill, 2009.

[2] D. Galin, *Software Quality: Concepts and Practice*. IEEE Computer Society, 2018.

[3] P. Thi, M. Phuong, and P. H. Thong, "Empirical Study of Software Defect Prediction : A Systematic Mapping," *Symmetry (Basel).*, Vol. 11, No. 212, Pp. 1–28, 2019. https://doi.org/10.3390/sym11020212

[4] T. Sethi and Gagandeep, "Improved Approach for Software Defect Prediction using Artificial Neural Networks," in *5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2016, Pp. 480–485. https://doi.org/10.1109/ICRITO.2016.7785003

[5] E. Irawan and R. S. Wahono, "Penggunaan Random Under Sampling untukPenangananKetidakseimbanganKelaspadaPrediksiCacat Software Berbasis Neural Network," *J. Softw. Eng.*, Vol. 1, No. 2, Pp. 92–100, 2015.

[6] X. Rong, F. Li, and Z. Cui, "A model for software defect prediction using support vector machine based on CBA," *Int. J. Intell. Syst. Technol. Appl.*, Vol. 15, No. 1, Pp. 19–34, 2016.

[7] S. A. Putri and R. S. Wahono, "Integrasi SMOTE dan Information Gain pada Naive Bayes untukPrediksiCacat Software," *J. Softw. Eng.*, Vol. 1, No. 2, 2015.

[8] J. Han and M. Kamber, *Data Mining: Concepts and Techniques Second Edition*. San Farnsisco: Elsevier Inc., 2006.

[9] G. Huang, Q. Zhu, and C. Siew, "Extreme Learning Machine : Theory and Applications," *Neurocomputing*, Vol. 70, Pp. 489–501, 2006. https://doi.org/10.1016/j.neucom.2005.12.126

[10] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme Learning Machine for Regression and Multiclass Classification," *IEEE Trans. Syst. Man, Cybern. - Part B Cybern.*, Vol. 42, No. 2, Pp. 513–528, 2012. https://doi.org/10.1109/TSMCB.2011.2168604

[11] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Syst. Appl.*, Vol. 73, Pp. 220–239, 2017. https://doi.org/10.1016/j.eswa.2016.12.035

[12] G. Douzas, F. Bacao, and F. Last, "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE," *Inf. Sci. (Ny).*, Vol. 465, Pp. 1–20, 2018. https://doi.org/10.1016/j.ins.2018.06.056

[13] N.V.Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell.Res.*, Vol. 16, Pp. 321–357, 2002. https://doi.org/10.1613/jair.953

[14] Y. Suh, J. Yu, J. Mo, L. Song, C. Kim, "A Comparison of Oversampling Methods on Imbalanced Topic Classification of Korean News Articles", *J. Cognitive Sci.*, Vol. 18, No. 4, Pp 391-437, 2017. https://doi.org/10.17791/jcs.2017.18.4.391

[15] P. Sarakit, T. Theeramunkong, and C. Haruechaiyasak, "Improving Emotion Classification in Imbalanced YouTube Dataset Using SMOTE Algorithm," in *2nd International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, 2015. https://doi.org/10.1109/ICAICTA.2015.7335373

[16] L. Demidova and I. Klyueva, "SVM Classification : Optimization with the SMOTE Algorithm for the Class Imbalance Problem," in *6th Mediterranean Conference on Embedded Computing (MECO)*, 2017, No. 11-15 June, Pp. 17–20. https://doi.org/10.1109/MECO.2017.7977136

[17] W. Zong, G. Bin Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, Vol. 101, Pp. 229–242, 2013. https://doi.org/10.1016/j.neucom.2012.08.010

[18] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data Quality : Some Comments on the NASA Software Defect Datasets," Vol. 39, No. 9, Pp. 1208–1215, 2013. https://doi.org/10.1109/TSE.2013.11

[19] G. Huang, "Extreme Learning Machine - Learning Without Iterative Tuning." Tutorial in IJCNN2012/WCCI2012, Brisbane, 2012.

[20] I. K. Timotius and S. G. Miaou, "Arithmetic Means of Accuracies: A Classifier Performance Measurement for Imbalanced Data Set," in *International Conference on Audio, Language and Image Processing (ICALIP)*, 2010, Pp. 1244–1251. https://doi.org/10.1109/ICALIP.2010.5685124

[21] S. Haykin, *Neural Networks - A Comprehensive Foundation, Second Edition*. India: Pearson Education, 2005.

[22] H. Rianto and R. S. Wahono, "Resampling Logistic Regression untukPenangananKetidakseimbangan Class padaPrediksiCacat Software," *J. Softw.Eng.*, Vol. 1, No. 1, Pp. 46–53, 2015.